IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received May 7, 2021, accepted May 18, 2021, date of publication June 11, 2021, date of current version June 23, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3088414

An Empirical Evaluation of Stacked Ensembles With Different Meta-Learners in Imbalanced Classification

SENG ZIAN[1], SAMEEM ABDUL KAREEM[1], AND KASTURI DEWI VARATHAN[2]
[1]Department of Artificial Intelligence, Faculty of Computer Science and Information Technology, Universiti Malaya, Kuala Lumpur 50603, Malaysia
[2]Department of Information Systems, Faculty of Computer Science and Information Technology, Universiti Malaya, Kuala Lumpur 50603, Malaysia

Corresponding author: Kasturi Dewi Varathan (kasturi@um.edu.my)

This work was supported by the Faculty Research Grant of Universiti Malaya under Grant GF011D-2019.

ABSTRACT The selection of a meta-learner determines the success of a stacked ensemble as the meta-learner is responsible for the final predictions of the stacked ensemble. Unfortunately, in imbalanced classification, selecting an appropriate and well-performing meta-learner of stacked ensemble is not straightforward as different meta-learners are advocated by different researchers. To investigate and identify a well-performing type of meta-learner in stacked ensemble for imbalanced classification, an experiment consisting of 19 meta-learners was conducted, detailed in this paper. Among the 19 meta-learners of stacked ensembles, a new weighted combination-based meta-learner that maximizes the H-measure during the training of stacked ensemble was first introduced and implemented in the empirical evaluation of this paper. The classification performances of stacked ensembles with 19 different meta-learners were recorded using both the area under the receiver operating characteristic curve (AUC) and H-measure (a metric that overcomes the deficiencies of the AUC). The weighted combination-based meta-learners of stacked ensembles have better classification performances on imbalanced datasets when compared to bagging-based, boosting-based, Decision Trees, Support Vector Machines, Naive Bayes, and Feedforward Neural Network meta-learners. Thus, the adoption of weighted combination-based meta-learners in stacked ensembles is recommended for their better performance on imbalanced datasets. Also, based on the empirical results, we identified better-performing meta-learners (such as the AUC maximizing meta-learner and the H-measure maximizing meta-learner) than the widely adopted meta-learner – Logistic Regression – in imbalanced classification.

INDEX TERMS Class imbalance, H-measure, imbalanced classification, meta-learner, stacked ensemble, stacking, super learning.

I. INTRODUCTION

A dataset is class imbalanced if the number of instances in one class is greatly outnumbered by another class [1], [2]. The classification task on a class imbalanced dataset is referred to as imbalanced classification. The topic of imbalanced classification is a valuable research hotspot in the literature [1]. The class with a higher number of instances is referred to as the majority class, whereas the class with fewer instances is recognized as the minority class. Imbalanced classification has many real-world applications such as early prediction of cardiac arrest in sepsis patients [3], software defect prediction [4], dropout prediction in massive open online courses (MOOCs) [5], etc. A poor performance model may result in serious consequences such as monetary loss and poor medical treatment timing. Thus, high-performance prediction models are anticipated for many real-world applications. Unfortunately, the classification task is subjected to the problem of performance degradation on imbalanced datasets. In order to tackle the problem of suboptimal performance in imbalanced classification, various classification methods, including ensemble learning approaches, are proposed among researchers.

Ensemble learning is also known as multiple classifier systems (MCS) [6]. MCS improves the performances of individual classifiers by combining several classifiers to obtain

The associate editor coordinating the review of this manuscript and approving it for publication was Zhan Bu.

a new classifier system that outperforms every single classifier [7]–[9]. In the literature [10], ensemble learning is categorized into homogeneous and heterogeneous ensembles. The homogeneous ensemble consists of classifiers from the same learning algorithm, whereas the heterogeneous ensemble has diverse classifiers from different learning algorithms. Since diversity is an important criterion for the success of an ensemble, the heterogeneous ensemble can improve the performance of an ensemble by promoting classifier diversity within the ensemble. In this paper, we are particularly interested in Stacked Generalization (SG) [11] or stacked ensemble, an ensemble that is commonly used to formulate a heterogeneous ensemble [9], [12].

The stacked ensemble, which is increasingly popular in imbalanced classification, has its peculiarity, i.e., a trainable meta-learner is required, compared to bagging and boosting. The final prediction of a stacked ensemble is equivalent to the output of its meta-learner [9], which means that the imbalanced classification performance of a stacked ensemble depends on the selection of its meta-learner. Based on the literature review of stacked ensembles in imbalanced classification (Section II-C), a variety of algorithms, including Logistic Regression, Decision Tree [13], Naive Bayes, Neural Network, Bagging [14], Support Vector Machine (SVM) [15], Random Forest [16], eXtreme Gradient Boosting (XGBoost) [17], Light Gradient Boosting Machine (LightGBM) [18], etc., are employed as the meta-learners of stacked ensembles. In other words, different researchers advocate different meta-learners for stacked ensemble in imbalanced classification (please refer to Table 1). In short, this research is motivated by the fact that the meta-learner is responsible for the success (i.e., imbalanced classification performance) of a stacked ensemble. However, the selection of meta-learner is not a straightforward task as different researchers promote different meta-learners. Hence, to select and configure the meta-learner of stacked ensemble based on knowledge and evidence of imbalanced classification, a number of meta-learners were empirically evaluated on imbalanced datasets in this paper.

Based on the literature review in Section II-E, the H-measure, which overcomes the deficiencies of the area under the receiver operating characteristic curve (AUC), was adopted as the evaluation metric in the empirical evaluation of meta-learners. However, considering that the AUC is widely adopted in imbalanced classification research, the AUC was also included, i.e., both the H-measure and AUC were used as the experiment's performance evaluation metrics. Inspired by the importance of H-measure in the literature (Section II-E), a new H-measure maximizing meta-learner of stacked ensemble was also implemented and evaluated in the experimental evaluation. This new H-measure maximizing meta-learner has never been explored and no published articles in this area can be found in the literature.

To the best of our knowledge, this research paper is the first empirical evaluation paper on the implementation and evaluation of the stacked ensemble with different meta-learners, including the H-measure maximizing meta-learner on imbalanced datasets. Furthermore, this paper is also the first imbalanced classification research that evaluates a wide range of meta-learners and reports their performance using both AUC and H-measure metrics. More importantly, the findings concluded from this research may support the researchers in terms of the selection of meta-learner in stacked ensemble for imbalanced classification.

The outline of this paper is organized as follows. Section II introduces the related literature about imbalanced classification problems, the stacked ensemble and its meta-learners in imbalanced classification, the machine learning algorithms utilized in base learning and meta-learning of stacked ensembles in the experiment, and the performance evaluation criteria. Section III describes the experimental settings, including the datasets used, the hyper-parameters of base learners and meta-learners in stacked ensembles. Section IV shows the experimental results and related descriptive and statistical analyses. Section V presents the conclusion.

II. LITERATURE REVIEW

This section reviews the literature of imbalanced classification from five main aspects, i.e., the problems and difficulties in imbalanced classification, the concept of stacked ensemble, stacked ensembles' meta-learners in imbalanced classification, the classification algorithms used in base learning and meta-learning of stacked ensembles in the empirical evaluation, and the performance evaluation criteria.

A. PROBLEMS AND DIFFICULTIES IN IMBALANCED CLASSIFICATION

The main problem of imbalanced classification is the suboptimal classification performance on imbalanced datasets. There noticed the issue of performance degradation for all degrees of class imbalance in imbalanced datasets [19]–[21]. Several difficulties, including standard classification algorithms and inappropriate global learning metrics, were responsible for the performance degradation in imbalanced classification [1], [21], [22].

Standard classification algorithms are designed based on the assumptions that the class distribution is balanced and the misclassification costs are equal [23]–[25]. When these standard classification algorithms are applied to imbalanced datasets, there is a bias towards the majority instances, leading to suboptimal classification performance. For example, in a standard rule learning algorithm, classification rules that classify the rare minority instances are often not generated because the minority instances are infrequently available in imbalanced datasets [23].

A metric can be used as the performance evaluation metric (the former) in the experimental evaluation and as the global learning metric (the latter) during the training of a classification algorithm. If an inappropriate metric is used as the experimental evaluation metric (the former), a misleading evaluation of classifiers can occur. Meanwhile, as reported in the literature [1], if an inappropriate metric is used as

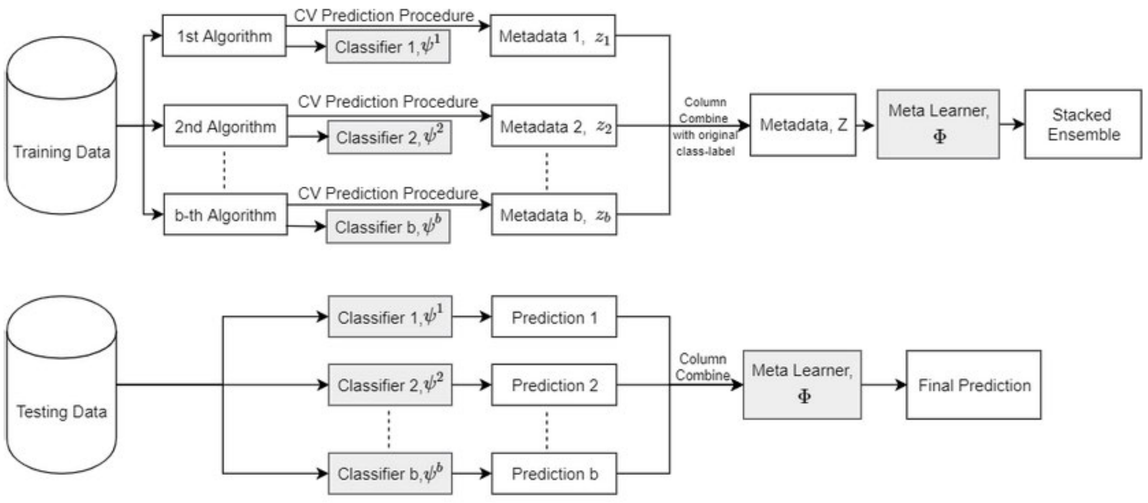

FIGURE 1. Framework of stacked ensemble.

the learning metric in the training of an algorithm (the latter), the classifier is misguided, leading to suboptimal classification performance. Both scenarios are undesirable. To consider the former, the selection of appropriate performance evaluation metrics is reviewed and discussed in Section II-E. Specifically, the literature reviewed highlights the adoption of H-measure. Inspired by the importance of H-measure in the literature (Section II-E) and to consider the latter, a new meta-learner, in which the H-measure is used as the global learning metric, is implemented and evaluated in this paper.

B. STACKED ENSEMBLE

Stacked Generalization [11] also known as stacking or stacked ensemble, is an ensemble that consists of multiple base learners and a meta-learner that learns and combines the predictions of base learners [9]. Stacked ensemble is considered a heterogeneous ensemble that promotes classifier diversity because the base learners of stacked ensemble are usually generated using different and diverse learning algorithms [9], [12]. Van der Laan *et al.* [26] proved the theoretical oracle property of the stacked ensemble and coined the algorithm as Super Learning or Super Learner. According to Naimi and Balzer [27], the Super Learner, an evolved version of Stacked Generalization, is an ensemble that consists of an optimal-weighted combination of base learners. The optimality is defined by a user-specified objective function or loss function [27], [28]. In the literature [29]–[31], Super Learning is recognized as Stacked Generalization and used interchangeably. In this paper, rather than using Stacked Generalization and Super Learning interchangeably, the term 'stacked ensemble' is used.

Fig. 1 shows the generic framework of a stacked ensemble. Suppose that given a training dataset with dimension of $n \times f$, where n is the number of rows and f is the number of features or columns, and b, the number of classification learning algorithms.

In the training step of Fig. 1, for a single b-th iteration, the given training dataset is used as the input for the b-th base algorithm, which goes on to perform two important tasks. Firstly, the training of b-th base classifier, ψ^b for the testing step. Specifically, the b-th classification algorithm learns the whole training dataset (without cross-validation) and generates the b-th base classifier of stacked ensemble. Secondly, the generation of metadata, z_b for each of the b-th classification algorithm. Fig. 2 shows the k-fold cross-validation method with $k = 5$ as an example. The k-fold cross-validation is a common approach to generate the metadata in stacked ensemble [9], [28]. Thus, in the experiment of this paper, the k-fold cross-validation method is adopted. Fig. 1 shows the generation of metadata, z_b using the k-fold cross-validation method. In Fig. 2, the training dataset is partitioned into k-roughly-equal folds, $\{F_1, \ldots, F_k\}$. For each fold, F_k, the out-of-k-th folds (gray-folds) are used to train a classifier, $\psi^b_{F_k}$ and predict on the fold F_k (slash-line-fold), repeated for k times, then the predicted folds $F_1, \ldots, F_k$ (slash-line-folds) are combined into a metadata of b-th algorithm, z_b (with a dimension of $n \times 1$). The two important tasks are repeated b times for b base algorithms. The generated z_b are column-combined along with the original actual class-label to complete the metadata, Z. The metadata, Z is used as the learning data for the different meta-learners, Φ (listed in Table 5) in the empirical evaluation of this paper.

In the testing step of the stacked ensemble, the testing data is used as input of the b base learners, $\{\psi^1, \ldots, \psi^b\}$. Predictions of the b base learners, $\{\psi^1, \ldots, \psi^b\}$ are column-combined and fed into the trained meta-learner, Φ. The prediction of meta-learner, $\hat{\Phi}$ is the final prediction of the stacked ensemble with meta-learner, Φ.

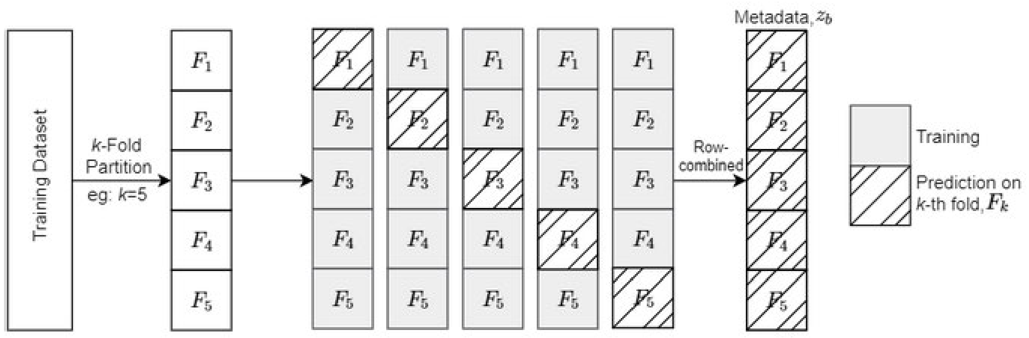

FIGURE 2. k-fold cross-validation for the generation of metadata, z_b, k = 5.

C. LITERATURE ON META-LEARNERS OF STACKED ENSEMBLE

In this section, research journals related to stacked ensembles in imbalanced classification are gathered and reviewed from the perspective of its meta-learner. In the literature, a wide range of meta-learners has been adopted for imbalanced classification.

The single classifier system, i.e., non-ensemble-based classifier, is employed as the meta-learner of stacked ensemble in the following imbalanced classification literature. Idris and Khan [32] compared the stacked ensemble against majority voting-based ensemble on telco datasets. The stacked ensemble consists of four base learners, i.e., Random Forest, Rotation Forest, RotBoost, and SVM, and a meta-learner of Decision Tree. The cross-validated predictions in class-label are used as the metadata of the stacked ensemble. They concluded that the majority voting-based ensemble outperformed stacking with class-label in churn prediction. However, they added that the performance of stacked ensemble could be improved by using predicted probability rather than the predicted class-label as metadata. Thus, in this paper, the metadata used in the empirical evaluation is the predicted probability. Ahmed *et al.* [33] investigated the integration of stacked ensemble in both bagging and boosting ensembles named Bagged-Stacked learner and Boosted-Stacked learner, respectively. The stacked ensemble with Decision Tree as a meta-learner is employed as the classifier within bagging and boosting. Therefore, both the Bagged-Stacked learner and Boosted-Stacked learner's final predictions are the votings of multiple stacked ensembles. Wijaya and Wahono [4], as well as Grenet *et al.* [34], applied stacked ensemble with Naive Bayes as meta-learner in software defect prediction and chemical risk evaluation, respectively. Grenet *et al.* [34] highlighted that the proposed stacked ensemble outperformed the industry's classical QSAR algorithms. Cao *et al.* [35] proposed a stacked ensemble classifier named lncLocator to predict the lncRNA subcellular localizations. They utilized Neural Network as the meta-learner in the stacked ensemble. Akhtar *et al.* [36] employed the stacked generalization method in the prediction of Large-For-Gestational-Age (LGA) fetus, i.e., the classification of a fetus as LGA or non-LGA. The cross-validated predictions of base learners were used as the metadata of SVM with linear kernel. The researchers [37] also employed SVM (with the radial kernel) as the meta-learner of stacked in identifying abnormal phone calls.

Some researchers investigated the use of MCS, i.e., ensemble learning instead of a single classifier as the stacked ensemble's meta-learner. Tozlu *et al.* [38] proposed the use of stacked ensemble with Random Forest as the meta-learner in commercial loan application prediction. Gomaa *et al.* [39] applied bagging as a meta-learner of stacked ensemble to categorize user reviews in mobile app stores. The researchers [40] investigated the use of bagging and AdaBoost ensembles in the meta-learning of stacked ensemble for software defect prediction. A stacked ensemble with LightGBM as a meta-learner was proposed in the sleep stage classification [41]. Xia *et al.* [42] proposed a heterogeneous stacked ensemble named bstacking, which combines the bagging and stacking algorithm. The average prediction of meta-learners (i.e., XGBoost) is computed as the final prediction. Zhou *et al.* [43] also investigated the application of stacked ensemble with XGBoost as the meta-learner in purchase prediction.

The adoption of a robust linear classifier with good interpretability as the meta-learner of stacked ensemble is highlighted in the literature. Several researchers [3], [30], [31], [44]–[50] employed Logistic Regression as the linear meta-learner of stacked ensembles. Logistic Regression is the most adopted meta-learner of stacked ensemble in imbalanced classification based on the literature review. LeDell *et al.* [28] proposed an AUC-maximizing stacked ensemble in which a meta-learner finds the optimal combination of base learners that maximize the AUC metric. The AUC-maximizing stacked ensemble is coined as a Super Learning ensemble. Table 1 summarized the meta-learners adopted in the imbalanced classification literature.

D. ALGORITHMS OF BASE LEARNERS AND META-LEARNERS IN EMPIRICAL EVALUATION

1) ALGORITHM OF BASE LEARNER

Stacked ensemble is a heterogeneous ensemble generated using different learning algorithms [9], [12]. Since this paper aims to evaluate the performance of different meta-learners

TABLE 1. Meta-learners of stacked ensembles in imbalanced classification.

Meta-learner	Literature
Logistic Regression	[3], [30], [31], [43]–[49]
AUC-maximizing	[28]
Decision Tree	[32], [33]
Naive Bayes	[4], [34]
Neural Network	[35]
SVM with linear kernel	[36]
SVM with radial kernel	[37]
Random Forest	[38]
Bagging	[39], [40]
AdaBoost	[40]
XGBoost	[42], [43]
LightGBM	[41]

in stacked ensembles, the base learners are fixed for valid comparison and evaluation. The classification learning algorithms used as base learners including k-Nearest Neighbor, Feedforward Neural Network, Random Forest, C4.5 Decision Tree, Naive Bayes, and XGBoost are reviewed in this section.

a: k-NEAREST NEIGHBOR (k-NN)

k-NN is recognized as an instance-based learning algorithm [51]. The main idea of k-NN is that data instances with similar characteristics are near each other. In classification, the class prediction of a testing instance is generated based on the k closest training instances. The similarity of instances is measured in distance functions such as the Euclidean distance.

b: FEEDFORWARD NEURAL NETWORK

Feedforward Neural Network [52] has three basic layers, namely the input, hidden, and output layers. Each layer comprises elements named neurons. The neurons in one layer are connected to the neurons in the next layer. The network is trained to map the input to output through the neurons in the layers. In Feedforward Neural Network, the weight of the inter-neuron connection is adjusted iteratively in a feedforward process to obtain a good model.

c: RANDOM FOREST

Random Forest [16] is a tree-based ensemble that integrates the bootstrap method and the random subspace method. The bootstrapping method samples the training dataset with replacement and generates multiple bags of data. For each bag, a subset of attributes is randomly selected using the random subspace method. Let the number of attributes in the selected subset of attributes be a. Then, the bag with the selected a attributes is used to train a tree model. The training process is repeated for all the bags. Later, the generated trees are subsequently combined to complete the Random Forest model.

d: C4.5 DECISION TREE

The C4.5 Decision Tree algorithm is a tree-based model developed by Quinlan [13]. It consists of internal nodes that split the input data into smaller partitions according to a discrete function of the input features and a series of leaf nodes to assign labels to data instances. The gain ratio is employed as the splitting criteria, and the splitting stops if the number of instances to be split is below a certain threshold. The C4.5 Decision Tree has good interpretability because it can be translated into a set of interpretable 'if…else' rules.

e: NAIVE BAYES

Naive Bayes is a probabilistic learning algorithm. It is designed based on Bayes' theorem and naively assumes that the attributes are conditionally independent. Suppose that given a class vector, y and n attributes vector, $\{x_1,\ldots, x_n\}$, the Bayes' theorem states that:

$$P(y|x_1,\ldots,x_n) = \frac{P(y)\cdot P(x_1,\ldots,x_n|y)}{P(x_1,\ldots,x_n)} \tag{1}$$

Using the naive conditional independence assumption of attributes, Eq. 1 is simplified to Eq. 2.

$$P(y|x_1,\ldots,x_n) = \frac{P(y)\cdot \prod_{i=1}^{n} P(x_i|y)}{P(x_1,\ldots,x_n)} \tag{2}$$

Notice that $P(x_1,\ldots,x_n)$ is a constant, therefore Eq. 2 can be written as Eq. 3.

$$P(y|x_1,\ldots,x_n) \propto P(y)\cdot \prod_{i=1}^{n} P(x_i|y) \tag{3}$$

A conditional probability model, $P(y|x_1,\ldots,x_n)$ which is recognized as the Naive Bayes learning algorithm, is obtained by solving the maximum argument of $P(y)\cdot \prod_{i=1}^{n} P(x_i|y)$.

f: EXTREME GRADIENT BOOSTING (XGBOOST)

XGBoost [17] is a tree-based boosting algorithm that merges multiple weak decision trees into a single strong model. XGBoost enhances the gradient boosting framework by incorporating Taylor's expansion [53] in loss function for fast approximation and injecting a regularization term in the loss function to control the complexity of trees.

2) ALGORITHM OF META-LEARNER

Classification algorithms utilized as stacked ensembles' meta-learners in the imbalanced classification literature are summarized in Table 1. These meta-learner algorithms are reviewed in this section. Notice that the Feedforward Neural Network, Random Forest, C4.5 Decision Tree, Naive Bayes, and XGBoost algorithms have been discussed in Section II-D1. Thus, the remaining meta-learners algorithms, including Logistic Regression, C5.0 Decision Tree, CART Decision Tree, SVM, Bagging, AdaBoost, LightGBM, Super Learning with NNLS, CCLL, and AUC maximizing methods are reviewed in the following paragraphs.

a: LOGISTIC REGRESSION

Logistic Regression is a parametric model that predicts a binary outcome from a set of independent variables. It models the log odds of the outcome (i.e., $log(\frac{p}{1-p})$) as a linear function, as shown in Eq. (4). By moving some terms around, Eq. (4) can be expressed as shown in Eq. (5). Notice that Eq. (5) is a sigmoid function parameterized by z as shown in Eq. (6), where z consists of coefficients, $\{\beta_0, \ldots, \beta_f\}$ and features, $\{x_1, \ldots, x_f\}$.

$$log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_f x_f \tag{4}$$

and

$$p = \frac{1}{1 + exp^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_f x_f)}} \tag{5}$$

where:

$p =$ the probability of an outcome
$\beta_0 =$ the intercept
$\beta_1, \ldots, \beta_f =$ the coefficients
$f =$ the number of independent variables (i.e., features)

$$Sigmoid(z) = \frac{1}{1 + exp^{-z}} \tag{6}$$

b: C5.0 DECISION TREE

The C5.0 algorithm is a successor to Quinlan's C4.5 algorithm. It has several improvements that are likely to generate smaller trees compared to the C4.5 algorithm [54]. For example, the C5.0 will combine non-occurring conditions for splits with several categories. Also, it performs a global pruning procedure that removes the sub-trees until the error rate exceeds one standard error of the baseline rate (i.e., no pruning). In terms of computing efficiency, the C5.0 algorithm is faster and requires less memory than the C4.5 algorithm.

c: CART DECISION TREE

The Classification and Regression Tree (CART) is a binary decision tree proposed by Breiman [55]. Beginning with the root node that contains all the training instances, the CART binary tree is constructed by splitting a node into two child nodes repeatedly. Different from C4.5 and C5.0 that used the gain ratio as the splitting criteria, the GINI index is adopted in the CART algorithm.

d: SUPPORT VECTOR MACHINE (SVM)

SVM [15] is also known as support-vector networks, which constructs a decision boundary that best separates (classify) the data instances. The best decision boundary of SVM is a separating hyperplane that has the largest distance to the nearest training instances of any class, i.e., a hyperplane that maximizes the margin. Boser *et al.* [56] extended the linear nature of the SVM model to non-linear classification using the 'kernel trick', i.e., replacing the simple linear cross product with the kernel function [54]. Table 2 presents the kernel equations of the kernel trick [54], [57], [58], where γ is the scaling parameter, *coef* is a coefficient, and d is the degree of the polynomial.

TABLE 2. Kernel equations of SVM.

Kernel	Equation
Linear	$K(x, y) = x \cdot y$
Polynomial	$K(x, y) = (\gamma x \cdot y + coef)^d$
Radial basis function	$K(x, y) = exp(-\gamma \|x - y\|^2)$
Sigmoid	$K(x, y) = tanh(\gamma x \cdot y + coef)$

e: BAGGING

Bootstrap aggregating (bagging) [14] is an ensemble learning method that generates multiple homogeneous classifiers using bootstrapping (i.e., random sampling with replacement), and these classifiers are aggregated to form the ensemble predictor. Suppose that given a training dataset, $\{x_N, y_N\}$, where N is the number of instances. The bagging algorithm generates b number of bags (each bag has size N) using random sampling with replacement. These b bags are used as the training datasets for a learning algorithm to generate b number of homogeneous classifiers, $\{c_1, c_2, \ldots, c_b\}$. Subsequently, $c_1, c_2, \ldots, c_b$ are aggregated by majority or weighted voting to construct the bagging ensemble.

f: ADABOOST

AdaBoost [59], short for Adaptive Boosting, is an ensemble learning method that iteratively learns from the mistakes of weak classifiers, and aggregates them (i.e., the weak classifiers) into a strong ensemble. Let I denote the total iterations in AdaBoost. In the i-th iteration of AdaBoost, a weak classifier, c_i is trained using a training dataset with weight distribution, D_i. The AdaBoost gives more focus to the difficult instances (i.e., incorrectly classified instances by c_i) by adjusting the weight distribution of training dataset in the next iteration, D_{i+1}. Specifically, the weights of difficult instances and easy instances are increased and decreased, respectively, in D_{i+1}. Furthermore, in each iteration, another weight (w_i) is assigned to each weak classifier (c_i) based on its performance, i.e., higher weights are given to more accurate classifiers. After all the I iterations, the I number of weak classifiers with the corresponding weights, $\{(c_1, w_1), \ldots, (c_i, w_i)\}$ are aggregated to form the AdaBoost ensemble.

g: LIGHT GRADIENT BOOSTING MACHINE (LIGHTGBM)

LIGHTGBM [18] is a new implementation of Gradient Boosting Decision Tree (GBDT) that consists of two new techniques, i.e., Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). The authors proved that the samples with larger gradients are crucial in the calculation of information gain. Thus, in the proposed GOSS, those instances with a small gradient are randomly removed, and those instances with a large gradient (i.e., under-trained instances) are used to estimate the information gain, i.e., estimation of the information gain with a smaller size of data using GOSS. Furthermore, in a sparse feature space, many features are mutually exclusive. In LightGBM, EFB is used to bundle these mutually exclusive features into a single feature (i.e., reduction of dimensionality) without compromising the model performance. The authors reported that the LightGBM is over 20 times faster in model training and has almost similar accuracy compared to conventional GBDT.

h: SUPER LEARNING WITH NNLS, CCLL, AND AUC-MAXIMIZING METHODS

The NNLS method is a meta-learner that finds the best-weighted combination (with non-negative-coefficient constraint) of base learners that minimizes the least squares error. The CCLL method is a meta-learner that searches the convex combination of base learners (i.e., the base learners are linearly combined, and their non-negative coefficients are summed to one) that minimizes the negative binomial log-likelihood on the logistic scale. The AUC-maximizing method is a meta-learner that finds the weighted combination of base learners that maximizes the AUC. It is worth mentioning that the Logistic Regression, NNLS method, CCLL method, and AUC-maximizing method find the weighted combination (linearly or non-linearly) of base learners with different loss functions, thus are considered as stacked ensembles with weighted combination-based meta-learners.

E. EVALUATION CRITERIA

Accuracy, commonly employed as the performance metric, is not a suitable metric in the imbalanced classification [1], [21], [60]. Consider an imbalanced credit card fraud dataset with 95 non-fraud instances (majority) and 5 fraud instances (minority). Recall that the minority class (i.e., fraud) is often the target of interest for a prediction model. A model that is trained to detect fraud instances is worthless if it fails to do so. Suppose that the trained credit card fraud classifier predicts all 100 instances as non-fraud (majority class), the accuracy of the classifier is 95%. Notice that all minority instances are misclassified in this 95% high accuracy model. Ironically, the fraud instance (minority class) is the target of the detection model. This shows that the accuracy metric is inappropriate as it is biased toward the majority class in the context of imbalanced classification [1].

On the other hand, the AUC metric, which is a popular performance evaluation metric in imbalanced classification [21], requires no input parameter including the threshold value, has the advantage of enabling reproducible evaluations among researchers. Although the AUC metric is generally accepted in the literature, Hand [61] highlighted the issue of incoherent misclassification costs in AUC, i.e., AUC assumes different misclassification cost distributions for different classification algorithms. Specifically, Hand [61] pointed out that the AUC is equivalent to averaging the misclassification loss over a cost ratio distribution, which depends on the classifiers' score distributions. In other words, the AUC metric is classifier-dependent, assumes different misclassification cost distributions for different classifiers. This is not logical because the misclassification cost should depend on the classification problem and not on the classifier [61], [62]. As an example, suppose that given 2 classifiers, $\{C_1, C_2\}$, the AUC utilizes the values of w and W as the misclassification cost of C_1 and C_2, respectively, where $w \neq W$. The incoherent misclassification cost is a serious problem because it means that the AUC evaluates different classifiers using different performance metrics [21], [61] and leads to unfair performance measurements. Therefore, the H-measure [61], which overcomes AUC's deficiencies, is proposed as an alternative performance metric.

H-measure is a metric that gives a normalized classifier assessment based on expected minimum misclassification loss [62]. A beta distribution is employed in H-measure to represent the relative cost distribution which is consistent across classifiers. The value of the H-measure ranges from 0 to 1. The value of 0 represents a random classifier, and 1 denotes a perfect classifier. The beta distribution, $beta(\pi_1+1, \pi_0+1)$ where π_0 and π_1 denote the proportions of instances belonging to class 0 and 1, respectively, is suggested for imbalanced classification [63]. In short, the H-measure is a suitable performance metric that can make valid comparisons among the classifiers in imbalanced classification, unlike the AUC metric, which measures different classifiers using different 'rulers'. In the literature, the authors [42], [62] utilized the H-measure as the performance metric in experimental evaluations. The H-measure can be estimated using Eq. 7 [61], as shown at the bottom of the next page.

with

$$\begin{aligned}\hat{L}_\beta &= \sum_{i=0}^{m}\{\pi_0(1-r_{0i})\left\{B(c_{(i+1)};1+\alpha,\beta)-B(c_{(i)};1+\alpha,\beta)\right\} \\ &\quad /B(1;\alpha,\beta) \\ &\quad +\pi_1 r_{1i}\left\{B(c_{(i+1)};\alpha,1+\beta)-B(c_{(i)};\alpha,1+\beta)\right\} \\ &\quad /B(1;\alpha,\beta)\} \qquad (8)\end{aligned}$$

$$c_{j+1} = \frac{\pi_1(r_{1(j+1)}-r_{1j})}{\pi_0(r_{0(j+1)}-r_{0j})+\pi_1(r_{1(j+1)}-r_{1j})} \qquad (9)$$

and

$$B(x;\alpha,\beta) = \int_0^x u^{\alpha-1}(1-u)^{\beta-1}du \qquad (10)$$

where:

(r_{1i}, r_{0i}) = the coordinates of ROC curve

π_0 = the proportions of instances belonging to class 0

π_1 = the proportions of instances belonging to class 1

m = the number of segments in the upper convex hull

$B(x;\alpha,\beta)$ = an incomplete beta function normalizing constant

Considering the widespread adoption of AUC as an evaluation metric in imbalanced classification and the importance of H-measure, both AUC and H-measure were employed as the evaluation metrics in this paper.

TABLE 3. Profile of experimental KEEL datasets.

Dataset	Real/Integer/Nominal	Total	IR	Majority Count	Minority Count	Minority Percentage
ecoli-0_vs_1	7 / 0 / 0	220	1.86	143	77	35.00%
iris0	4 / 0 / 0	150	2.00	100	50	33.33%
ecoli3	7 / 0 / 0	336	8.60	301	35	10.42%
yeast-0-2-5-7-9_vs_3-6-8	8 / 0 / 0	1004	9.14	905	99	9.86%
glass-0-4_vs_5	9 / 0 / 0	92	9.22	83	9	9.78%
led7digit-0-2-4-5-6-7-8-9_vs_1	7 / 0 / 0	443	10.97	406	37	8.35%
glass-0-6_vs_5	9 / 0 / 0	108	11.00	99	9	8.33%
shuttle-c0-vs-c4	0 / 9 / 0	1829	13.87	1706	123	6.72%
dermatology-6	0 / 34 / 0	358	16.90	338	20	5.59%
shuttle-c2-vs-c4	0 / 9 / 0	129	20.50	123	6	4.65%
shuttle-6_vs_2-3	0 / 9 / 0	230	22.00	220	10	4.35%
flare-F	0 / 0 / 11	1066	23.79	1023	43	4.03%
car-good	0 / 0 / 6	1728	24.04	1659	69	3.99%
yeast-1-2-8-9_vs_7	8 / 0 / 0	947	30.57	917	30	3.17%
abalone-3_vs_11	7 / 0 / 1	502	32.47	487	15	2.99%
winequality-white-9_vs_4	11 / 0 / 0	168	32.60	163	5	2.98%
yeast6	8 / 0 / 0	1484	41.40	1449	35	2.36%
shuttle-2_vs_5	0 / 9 / 0	3316	66.67	3267	49	1.48%
kr-vs-k-zero_vs_fifteen	0 / 0 / 6	2193	80.22	2166	27	1.23%
abalone19	7 / 0 / 1	4174	129.44	4142	32	0.77%

III. EXPERIMENTAL SETUP

This section presents the design of the experiment of this research, which includes the imbalanced datasets, the programming environment and settings, and the hyper-parameters of base learners and meta-learners used in the stacked ensembles.

A. IMBALANCED DATASETS

In order to evaluate stacked ensembles' performance with different meta-learners, 20 imbalanced datasets with different values of Imbalanced Ratio (IR) from various domains were used, as shown in Table 3. The datasets have different features, including numerical (real number and integer) and categorical (nominal) attributes. Specifically, Table 3 shows the profile of each dataset, including the type of features, the total number of instances, Imbalanced Ratio (IR), count of majority and minority instances, and the percentage of minority instances. These datasets are available online in the KEEL repository [64]. The datasets that are pre-partitioned into five folds by the KEEL repository were used to enable reproducible results. For every single 5-fold pre-partitioned imbalanced dataset, 10 data files were downloaded. With the "iris0" dataset as an example, the 5-fold pre-partitioned "iris0" dataset consists of 5 training datasets ("iris0-5-1tra.dat", "iris0-5-2tra.dat", "iris0-5-3tra.dat", "iris0-5-4tra.dat", and "iris0-5-5tra.dat") and 5 testing datasets ("iris0-5-1tst.dat", "iris0-5-2tst.dat", "iris0-5-3tst.dat", "iris0-5-4tst.dat", and "iris0-5-5tst.dat") as shown in Diagram 3. Thus, the "iris0" dataset is pre-partitioned in the KEEL repository into five folds, with each fold consisting of a training set and a testing set.

B. ENVIRONMENT AND SETTING

R version 3.6.3 [65] was used as the programming environment. Following the experimental setting in the literature [66], [67], this paper adopted the $N \times k$-fold cross-validation strategy where N is the number of repetition of the k-fold cross-validation and k is the number of folds in cross-validation. Specifically, the value of k is five since the 5-fold pre-partitioned KEEL datasets were used, and the value of N was set at five.

Fig. 3 shows the research experimental settings in the R programming environment, $N \times k$-fold cross-validation method where $N = 5$ and $k = 5$ as strategy, and "iris0" as the imbalanced dataset. For a single N-th iteration, and for each k-th fold, the training set was used to train a model (i.e., stacked ensemble) and the trained model predicts on the testing set to obtain the "performance on k-th fold". Repeated k times, k number of "performance on k-th fold" were produced and averaged to obtain the "averaged k-fold performance" for the single N-th iteration. The "averaged k-fold performance" produced by five iterations of N were averaged to obtain the final performances (in AUC and H-measure) of the experiment. In other words, the "final performance" was obtained by averaging the performance values of "averaged k-fold performance" for five repetitions ($N = 5$).

The training and testing steps of stacked ensembles are described in Section II-B and shown in Fig. 1. Specifically, the training dataset (e.g., "iris0-5-1tra.dat") was used as the

$$\hat{H} = 1 - \left(\frac{\hat{L}_\beta \cdot B(1; \alpha, \beta)}{\pi_0 B(\pi_1; 1+\alpha, \beta) + \pi_1 B(1; \alpha, 1+\beta) - \pi_1 B(\pi_1; \alpha, 1+\beta)}\right) \tag{7}$$

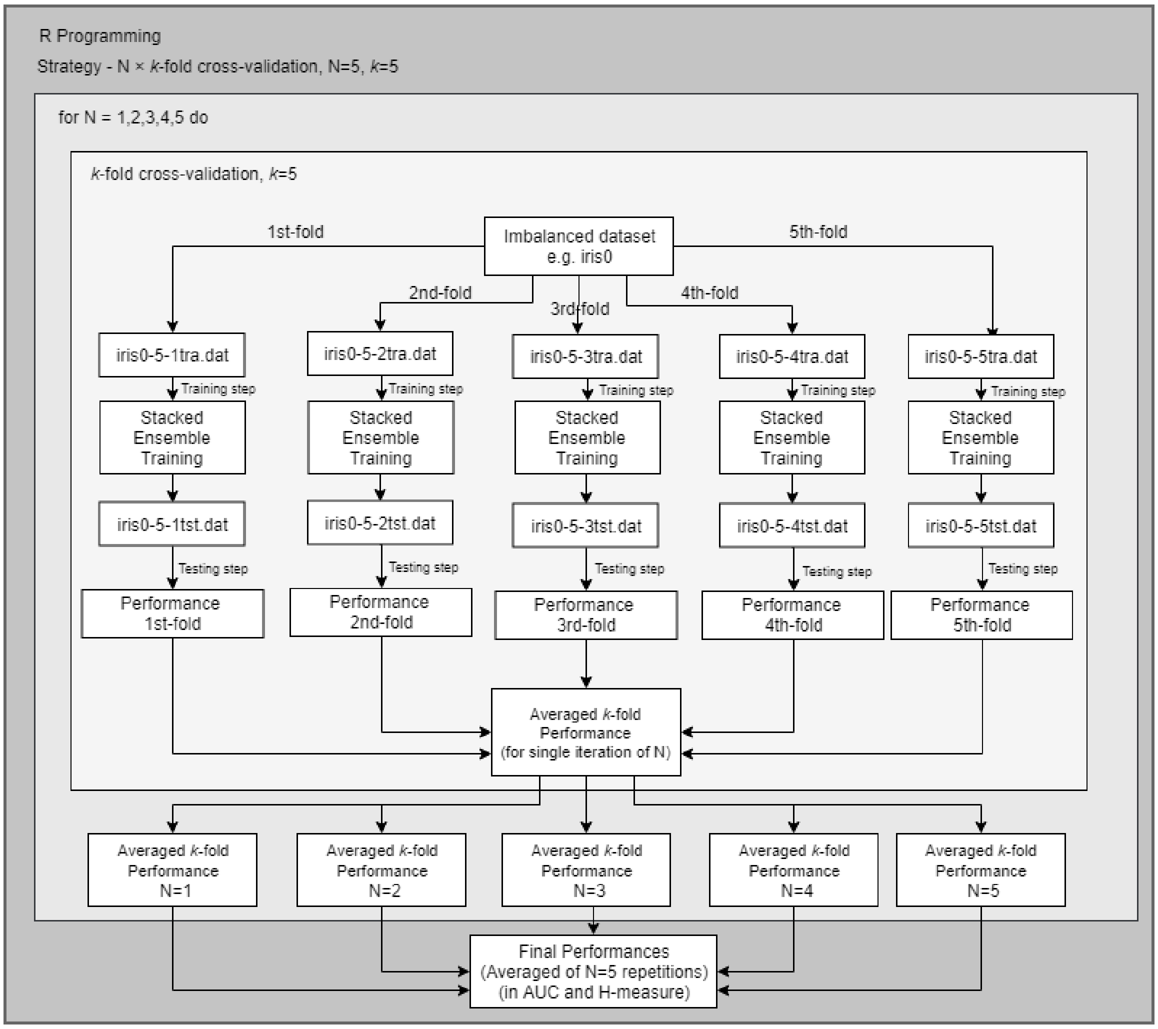
R Programming
Strategy - N × k-fold cross-validation, N=5, k=5
for N = 1,2,3,4,5 do
k-fold cross-validation, k=5
Imbalanced dataset e.g. iris0
1st-fold
2nd-fold
3rd-fold
4th-fold
5th-fold
iris0-5-1tra.dat
iris0-5-2tra.dat
iris0-5-3tra.dat
iris0-5-4tra.dat
iris0-5-5tra.dat
Training step
Stacked Ensemble Training
iris0-5-1tst.dat
iris0-5-2tst.dat
iris0-5-3tst.dat
iris0-5-4tst.dat
iris0-5-5tst.dat
Testing step
Performance 1st-fold
Performance 2nd-fold
Performance 3rd-fold
Performance 4th-fold
Performance 5th-fold
Averaged k-fold Performance (for single iteration of N)
Averaged k-fold Performance N=1
Averaged k-fold Performance N=2
Averaged k-fold Performance N=3
Averaged k-fold Performance N=4
Averaged k-fold Performance N=5
Final Performances (Averaged of N=5 repetitions) (in AUC and H-measure)

FIGURE 3. Diagram of experimental design - Overview.

input of b number of base learners (as listed in Table 4) to train the corresponding base classifiers. The training dataset was also subjected to the k-fold cross-validated prediction process to generate the metadata, Z (as described in Section II-B). Subsequently, the metadata, Z was used as the input data for the training of different meta-learners (as listed in Table 5). Note that the generation of metadata and training of base learners were conducted only once for different meta-learners, i.e., the realization of fixed base learners in all stacked ensembles with different meta-learners. For the testing step, the testing data (e.g., "iris0-5-1tst.dat") was predicted by the trained base learners. Then, the predictions of base learners were used as the input for a trained meta-learner to output the final prediction of stacked ensemble. Note that the final predictions of stacked ensemble were evaluated using AUC and H-measure in the experiment.

C. HYPER-PARAMETER OF STACKED ENSEMBLE

1) HYPER-PARAMETERS OF BASE LEARNERS

k-Nearest Neighbor, Feedforward Neural Network [52], Random Forest [16], eXtreme Gradient Boosting [17], C4.5 Decision Tree [13], and Naive Bayes were used as the base learners of stacked ensembles. The reasons for selecting these heterogeneous algorithms are two-fold. Firstly, a stacked ensemble is usually generated using different and diverse learning algorithms [9], [12]. In other words, a stacked ensemble is commonly a heterogeneous ensemble. Secondly, these heterogeneous algorithms are built on different principles and likely to make complementary errors [68], [69]. Table 4 shows the list of algorithms used as base learners, the hyper-parameters of base learners, and R packages utilized.

In Table 4, the base learners are parameterized using (mostly) default hyper-parameters rather than tuned hyper-parameters. Ultimately, a different set of base learners' hyper-parameters produces different metadata (which is the input data for the learning of meta-learner). Let the metadata generated using the base learners' hyper-parameters in Table 4 and a set of tuned base learners' hyper-parameters be denoted as Z_{used} and Z_{tuned}, respectively. Z_{tuned} consists of optimal base learners' predictions as it is generated based on tuned hyper-parameters of base learners,

TABLE 4. List of base learners of stacked ensembles in experiment.

Abbreviation	Base learner algorithm	Hyper-parameter	R package
nb	Naive Bayes	Default parameters	e1071 [58]
c45	C4.5 Decision Tree	Default parameters	RWeka [70]
k-NN	*k*-Nearest Neighbor	$k = 5$ and other default parameters	KernelKnn [71]
nnet	Feedforward Neural Network	maxit = 500, size = 2, trace = F, and other default parameters	nnet [52]
rf	Random Forest	ntree = 1000 and other default parameters	randomForest [72]
xgb	eXtreme Gradient Boosting	nrounds = 1000, max_depth = 4, eta = 0.1, min_child_weight = 10, and other default parameters	xgboost [73]

whereas Z_{used} contains a mixture of optimal and suboptimal base learners' predictions as it is generated based on less-tuned hyper-parameters of base learners. The learning of a meta-learner using Z_{used} (as input data) is more challenging than Z_{tuned} as the meta-learner needs to learn based on less-optimized input data. Thus, Z_{used} is preferred over Z_{tuned} in the experimental setting of this research. The main reason is that Z_{used} provides a more complex and challenging environment than Z_{tuned} for the learning (evaluation) of meta-learner. In other words, in this experiment, the meta-learners are tested in a 'difficult' environment rather than an 'easy' (tuned) environment to test the capability of meta-learners in merging the predictions of base learners (metadata). A meta-learner must overcome the difficulty in the given metadata to achieve good performance in the experiment. Thus, the current setting (Table 4) has the advantage of providing a more challenging environment using Z_{tuned} as the input learning metadata (of meta-learners) for the evaluation of meta-learners.

2) HYPER-PARAMETERS OF META-LEARNERS

The meta-learners reported in the literature (as shown in Table 1) are included in the empirical evaluation, as shown in Table 5. Their respective implementation in R programming and the hyper-parameters used in the empirical evaluation are highlighted. Table 5 includes not only those specifically reported meta-learners but also those meta-learners relevant to the reported meta-learners. C4.5, C5.0, and CART are included for the reported decision tree meta-learner; AUC-maximizing, NNLS, CCLL, and a new method named 'H-measure maximizing meta-learner' are covered for the reported meta-learners under the Super Learning framework. Since the new H-measure maximizing meta-learner, which has never been investigated in the literature (to the best of our knowledge), is first implemented in this paper, the following paragraph briefly describes its implementation.

3) IMPLEMENTATION OF A NEW H-MEASURE MAXIMIZING META-LEARNER

The main idea of the H-measure maximizing meta-learner is that the meta-learner learns the best-weighted combination of base learners that maximizes the cross-validated H-measure based on the base learners' prediction. Following the notations used in Section II-B, the implementation and development of the H-measure maximizing meta-learner is described. Specifically, in the development of the H-measure maximizing meta-learner, two major components, the optimization component and the H-measure calculation component were considered. For the optimization component, the nonlinear optimization package – nloptr was utilized to learn the optimal-weighted combination of base learners, $\{\psi^1, \ldots, \psi^b\}$ that maximizes the cross-validated H-measure on the metadata, Z. The nloptr package is an R interface to the NLopt [74] project from MIT and has several different effective optimization routines. For the H-measure calculation component, the 'hmeasure' package [75] in R programming was adopted. Following the recommendation for imbalanced data in the literature [63], the H-measure metric with parameters of beta distribution, $beta(\pi_1 + 1, \pi_0 + 1)$ was employed in the H-measure maximizing meta-learner. By integrating the H-measure as loss function and optimization algorithms in the NLopt project, a meta-learner, Φ, which finds the best combination of base learners, $\{\psi^1, \ldots, \psi^b\}$, is formulated.

D. PERFORMANCE ANALYSIS

Both descriptive statistics and statistical tests were adopted for classification performance analyses. The average performance and average ranking are calculated for descriptive statistics. Furthermore, to analyze the performances of stacked ensembles with different meta-learners from a statistical perspective, a hierarchical analysis that consists of intra-family and inter-family comparison was adopted [82].

For the pair-wise comparison of two methods, the non-parametric Wilcoxon signed-rank test was utilized [82], [83]. The null hypothesis of the non-parametric Wilcoxon signed-rank test is that the two methods have equal classification performances. If the non-parametric Wilcoxon signed-rank test's null hypothesis is rejected, it indicates that one of the methods has a statistically better classification performance than the other method.

TABLE 5. List of meta-learners of stacked ensembles in experiment.

No	Abbreviation	Type	Meta-learner	Literature	R Package	Hyper-parameter
1	SL-H	Weighted Combination	Meta-learner finds the weighted combination of base learners based on the H-measure maximization.	New meta-learner implemented in this paper	SuperLearner [76]	Default parameters
2	SL-AUC	Weighted Combination	Meta-learner finds the weighted combination of base learners based on the AUC maximization.	[28]	SuperLearner [76]	Default parameters
3	SL-CCLL	Weighted Combination	Meta-learner finds the convex combination (CC) of base learners based on the minimization of negative binomial log-likelihood (LL) on the logistic scale.	Evaluated in [28]	SuperLearner [76]	Default parameters
4	SL-NNLS	Weighted Combination	Meta-learner finds the weighted combination (with non-negative-coefficient constraint) of base learners based on the least squares error minimization.	Evaluated in [28]	SuperLearner [76]	Default parameters
5	SG-LR	Weighted Combination	Logistic Regression	[3], [30], [31], [44]–[49]	glm function in R [65]	Default parameters
6	SG-C45	Non-ensemble - Tree-based	Decision Tree	[32], [33]	RWeka [70]	Default parameters
7	SG-C50	Non-ensemble - Tree-based	Decision Tree	[32], [33]	C50 [77]	Default parameters
8	SG-CART	Non-ensemble - Tree-based	Decision Tree	[32], [33]	rpart [78]	Default parameters
9	SG-NB	Non-ensemble - Bayesian	Naive Bayes	[4], [34]	e1071 [58]	Default parameters
10	SG-NNET	Non-ensemble - Neural Network	Neural Network	[35]	nnet [52]	maxit = 500, size = 2, trace = F, and other default parameters
11	SG-SVM-Linear	Non-ensemble - SVM	SVM with linear kernel	[36]	e1071 [58]	Default parameters
12	SG-SVM-Poly	Non-ensemble - SVM	SVM with polynomial kernel	Kernel variant of reported SVM [36], [37]	e1071 [58]	Default parameters
13	SG-SVM-Radial	Non-ensemble - SVM	SVM with radial basis kernel	[37]	e1071 [58]	Default parameters
14	SG-SVM-Sigmoid	Non-ensemble - SVM	SVM with sigmoid kernel	Kernel variant of reported SVM [36], [37]	e1071 [58]	Default parameters
15	SG-Bag	Ensemble - Bagging	Bagging	[39], [40]	ipred [79]	Default parameters
16	SG-RF	Ensemble - Bagging	Random Forest	[38]	randomForest [72]	ntree = 1000 and other default parameters
17	SG-Ada	Ensemble - Boosting	AdaBoost	[40]	adabag [80]	Default parameters
18	SG-XGB	Ensemble - Boosting	eXtreme Gradient Boosting	[42], [43]	xgboost [73]	nrounds = 1000, max_depth = 4, eta = 0.1, min_child_weight = 10, and other default parameters
19	SG-LGBM	Ensemble - Boosting	LightGBM	[41]	lightgbm [81]	objective = binary and other default parameters

For comparisons of multiple methods, the Friedman Aligned Ranks test was first used to examine if there is a statistically significant difference among the methods [84]. The null hypothesis of the Friedman Aligned Ranks test is that all the methods have equal classification performances. If the null hypothesis of the Friedman Aligned Ranks test is rejected, it indicates unequal classification performances among the methods. Since unequal classification performances among the methods exist, to further analyze and compare the performances among the methods, a post hoc test with p-values adjusted using Bergman and Hommel's procedure is performed. On the other hand, no post hoc test is conducted if the Friedman Aligned Ranks test indicates that there is no significant difference among the methods (i.e., Friedman Aligned Ranks test's null hypothesis is not rejected).

IV. EMPIRICAL CLASSIFICATION PERFORMANCE RESULT

A. CLASSIFICATION PERFORMANCE ANALYSIS

This section shows the classification performance results on the KEEL imbalanced datasets. In this experiment, the meta-learners of stacked ensembles were categorized into non-ensemble, ensemble, and weighted combination, as shown in Table 5.

For each of the evaluated stacked ensembles with different meta-learner, the average H-measure and average AUC were calculated by averaging the recorded H-measure and AUC metrics over all the KEEL datasets. Besides, the average rank was calculated for each stacked ensemble by averaging the fractional ranking of the 19 evaluated stacked ensembles with different meta-learners over the KEEL datasets. The average AUC and average rank (based on AUC) were computed as shown in Table 6, whereas the average H-measure and average rank (based on H-measure) were calculated as presented in Table 7.

From the average H-measure perspective, both SL-H and SL-CCLL have the highest average H-measure of 0.857, followed by SL-AUC with a value of 0.854. For average rank based on H-measure, the newly implemented stacked ensemble with H-measure maximizing meta-learner (SL-H) has the best average rank of 5.85 and is followed by SL-AUC with the value of 5.88. On the other hand, the SL-CCLL has the highest average AUC (0.9530), as shown in Table 6. For the average rank based on the AUC metric, the SL-AUC recorded the best average rank of 5.60, followed by SL-H with 5.88. There observed good classification performances in stacked ensembles with weighted combination-based meta-learners (i.e., SL-AUC, SL-CCLL, and SL-H) with both the AUC and H-measure as evaluation metrics.

B. STATISTICAL CLASSIFICATION PERFORMANCE ANALYSIS

As shown in Table 5, the types of meta-learner in stacked ensembles can be grouped into non-ensemble-based, weighted combination-based, and ensemble-based meta-learners. A hierarchical analysis [82], which consists of intra-group and inter-group comparisons, is adopted in this paper's statistical analysis. Specifically, three intra-group comparisons, i.e., comparisons within non-ensemble-based, weighted combination-based, and ensemble-based meta-learners, were first conducted and followed by an inter-group comparison. The inter-group comparison includes the top-performing stacked ensembles from each intra-group, i.e., comparison among stacked ensembles with non-ensemble-based, weighted combination-based, and ensemble-based meta-learners.

The diagram of hierarchical analysis is shown in Fig. 4. The double-straight-dotted line refers to the selected stacked ensemble's meta-learner. The single-straight-dotted line indicates no statistical significance attained by a stacked

TABLE 6. Performance in AUC: average rank and average AUC.

	Weighted Combination					Decision Tree			Bayesian	Neural Net	SVM				Bagging		Boosting		
Dataset	SL-H	SL-AUC	SL-CCLL	SL-NNLS	SG-LR	SG-C45	SG-C50	SG-CART	SG-NB	SG-NNET	SG-SVM-Linear	SG-SVM-Poly	SG-SVM-Radial	SG-SVM-Sigmoid	SG-Bag	SG-RF	SG-Ada	SG-XGB	SG-LGBM
ecoli-0_vs_1	0.992	0.992	0.990	0.990	0.973	0.977	0.974	0.985	0.987	0.985	**0.992**	0.992	0.980	0.992	0.986	0.986	0.988	0.987	0.500
iris0	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.500
ecoli3	0.940	0.938	0.937	0.934	0.941	0.772	0.783	0.810	**0.948**	0.500	0.933	0.937	0.915	0.936	0.935	0.942	0.910	0.939	0.534
yeast-0-2-5-7-9_vs_3-6-8	0.952	0.956	**0.962**	0.961	0.953	0.880	0.888	0.891	0.947	0.500	0.945	0.951	0.933	0.960	0.925	0.939	0.925	0.940	0.506
glass-0-4_vs_5	**1.000**	**1.000**	0.994	0.994	0.993	0.994	0.994	0.998	**1.000**	0.565	**1.000**	**1.000**	**1.000**	0.982	**1.000**	**1.000**	**1.000**	0.500	0.524
led7digit-0-2-4-5-6-7-8-9_vs_1	0.954	0.953	0.949	0.955	0.929	0.897	0.901	0.904	0.955	0.500	0.956	**0.956**	0.955	0.956	0.919	0.949	0.943	0.938	0.534
glass-0-6_vs_5	0.999	**1.000**	0.996	0.996	0.762	0.994	0.994	0.994	0.839	0.500	0.853	0.817	0.775	0.946	**1.000**	**1.000**	**1.000**	0.818	0.500
shuttle-c0-vs-c4	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	1.000	1.000	0.996	0.999	0.500	**1.000**	**1.000**	**1.000**	**1.000**	0.996	**1.000**	0.996	1.000	0.546
dermatology-6	**1.000**	**1.000**	0.999	0.999	0.999	0.999	0.999	**1.000**	0.999	0.500	**1.000**	**1.000**	**1.000**	0.997	**1.000**	**1.000**	**1.000**	0.985	0.521
shuttle-c2-vs-c4	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.996	0.998	0.931	0.994	0.500	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.643	0.500
shuttle-6_vs_2-3	**1.000**	**1.000**	0.998	0.998	0.998	0.988	0.988	**1.000**	**1.000**	0.500	**1.000**	**1.000**	**1.000**	0.992	**1.000**	**1.000**	**1.000**	0.993	0.520
flare-F	0.891	0.895	**0.898**	0.890	0.875	0.551	0.538	0.676	0.890	0.500	0.572	0.737	0.720	0.517	0.788	0.860	0.866	0.871	0.527
car-good	0.990	0.993	0.991	0.993	0.992	0.953	0.950	0.965	0.988	0.500	0.993	0.992	0.993	0.993	0.989	**0.995**	0.994	0.992	0.588
yeast-1-2-8-9_vs_7	0.749	0.736	0.768	0.761	0.674	0.541	0.558	0.542	**0.782**	0.500	0.513	0.537	0.503	0.544	0.650	0.662	0.616	0.707	0.501
abalone-3_vs_11	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.998	0.999	**1.000**	0.973	0.500	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.967	0.520
winequality-white-9_vs_4	0.868	0.868	0.866	0.871	0.607	0.579	0.625	0.505	0.873	0.500	0.507	0.581	0.540	0.529	**0.880**	0.856	0.862	0.500	0.520
yeast6	0.935	0.932	0.944	0.938	**0.945**	0.806	0.774	0.738	0.943	0.500	0.906	0.837	0.844	0.919	0.868	0.918	0.893	0.922	0.507
shuttle-2_vs_5	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.999	0.500	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.540
kr-vs-k-zero_vs_fifteen	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	1.000	**1.000**	0.984	0.500	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.999	0.511
abalone19	0.751	0.757	**0.771**	0.749	0.720	0.500	0.500	0.500	0.685	0.500	0.567	0.523	0.670	0.684	0.539	0.634	0.515	0.716	0.504
Average AUC	0.951	0.951	**0.953**	0.952	0.918	0.871	0.873	0.872	0.939	0.552	0.887	0.893	0.891	0.897	0.924	0.937	0.925	0.871	0.520
Average Rank	5.88	**5.60**	6.80	6.85	8.95	13.68	13.75	12.22	8.97	18.02	8.35	8.22	9.35	9.30	9.00	7.08	8.55	11.62	17.80

TABLE 7. Performance in H-measure: average rank and average H-measure.

	Weighted Combination					Decision Tree			Bayesian	Neural Net	SVM				Bagging		Boosting		
Dataset	SL-H	SL-AUC	SL-CCLL	SL-NNLS	SG-LR	SG-C45	SG-C50	SG-CART	SG-NB	SG-NNET	SG-SVM-Linear	SG-SVM-Poly	SG-SVM-Radial	SG-SVM-Sigmoid	SG-Bag	SG-RF	SG-Ada	SG-XGB	SG-LGBM
ecoli-0_vs_1	0.965	0.965	0.965	0.965	0.944	0.933	0.923	0.956	0.946	0.959	**0.966**	0.966	0.964	0.965	0.964	0.964	0.964	0.957	0.000
iris0	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.000
ecoli3	0.777	0.765	0.761	0.737	0.750	0.492	0.490	0.519	**0.792**	0.000	0.730	0.764	0.727	0.753	0.735	0.748	0.677	0.754	0.088
yeast-0-2-5-7-9_vs_3-6-8	0.803	0.794	0.802	0.801	0.801	0.733	0.739	0.736	0.809	0.000	0.794	**0.812**	0.790	0.803	0.788	0.791	0.775	0.794	0.100
glass-0-4_vs_5	**1.000**	**1.000**	0.975	0.975	0.969	0.975	0.975	0.990	**1.000**	0.123	**1.000**	**1.000**	**1.000**	0.950	**1.000**	**1.000**	**1.000**	0.000	0.000
led7digit-0-2-4-5-6-7-8-9_vs_1	0.798	0.799	0.798	0.805	0.791	0.755	0.757	0.765	0.798	0.000	0.802	0.800	0.802	0.802	0.788	**0.808**	0.798	0.790	0.059
glass-0-6_vs_5	0.996	**1.000**	0.983	0.983	0.653	0.974	0.974	0.974	0.900	0.000	0.910	0.841	0.431	0.944	**1.000**	**1.000**	**1.000**	0.382	0.000
shuttle-c0-vs-c4	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.999	0.998	0.991	0.996	0.000	**1.000**	**1.000**	**1.000**	**1.000**	0.991	**1.000**	0.991	0.999	0.451
dermatology-6	**1.000**	**1.000**	0.994	0.994	0.994	0.994	0.994	**1.000**	0.994	0.000	**1.000**	**1.000**	**1.000**	0.994	**1.000**	**1.000**	**1.000**	0.933	0.203
shuttle-c2-vs-c4	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.983	0.990	0.842	0.983	0.000	**1.000**	**1.000**	**1.000**	0.954	**1.000**	**1.000**	**1.000**	0.097	0.000
shuttle-6_vs_2-3	**1.000**	**1.000**	0.990	0.990	0.990	0.968	0.968	**1.000**	**1.000**	0.000	**1.000**	**1.000**	**1.000**	0.980	**1.000**	**1.000**	**1.000**	0.970	0.040
flare-F	0.594	0.579	**0.595**	0.578	0.588	0.076	0.056	0.296	0.594	0.000	0.467	0.490	0.422	0.311	0.451	0.520	0.519	0.536	0.058
car-good	0.926	0.934	0.939	0.934	0.927	0.863	0.848	0.870	0.923	0.000	0.931	0.935	0.932	0.933	0.935	**0.962**	0.955	0.939	0.160
yeast-1-2-8-9_vs_7	0.383	0.356	**0.403**	0.395	0.368	0.179	0.135	0.063	0.388	0.000	0.297	0.331	0.222	0.268	0.284	0.295	0.281	0.355	0.049
abalone-3_vs_11	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.991	0.995	**1.000**	0.967	0.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.854	0.120
winequality-white-9_vs_4	0.822	0.818	0.812	0.817	0.781	0.238	0.327	0.002	0.808	0.000	**0.836**	0.822	0.793	0.828	0.777	0.772	0.759	0.000	0.040
yeast6	0.763	0.754	**0.783**	0.766	0.781	0.558	0.523	0.429	0.766	0.000	0.743	0.719	0.713	0.754	0.666	0.715	0.663	0.757	0.172
shuttle-2_vs_5	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.997	0.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.559
kr-vs-k-zero_vs_fifteen	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.999	**1.000**	0.981	0.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.993	0.249
abalone19	0.303	0.315	**0.340**	0.304	0.292	0.000	0.000	0.000	0.264	0.000	0.263	0.255	0.293	0.242	0.074	0.174	0.193	0.292	0.084
Average H-measure	**0.857**	0.854	**0.857**	0.852	0.831	0.735	0.735	0.722	0.845	0.104	0.837	0.837	0.804	0.824	0.823	0.837	0.829	0.670	0.122
Average Rank	**5.85**	5.88	6.42	7.08	9.07	13.97	14.4	12.62	9.38	17.93	7.42	6.67	8.82	8.95	9.43	7.35	9.07	11.85	17.82

ensemble's meta-learner when compared to the selected meta-learner. The single-straight line refers to a statistical significance attained by a meta-learner when compared to the selected meta-learner.

1) INTRA COMPARISON OF STACKED ENSEMBLES WITH NON-ENSEMBLE-BASED META-LEARNERS (INTRA NON-ENSEMBLE GROUP)

As shown in Table 5, the non-ensemble-based meta-learners consists of different classification algorithms, including three decision trees (C4.5, C5.0, and CART), four SVMs with different kernels (SVM with linear, radial basis, polynomial, and sigmoid kernels), Naive Bayes algorithm, and Feedforward Neural Network algorithm. Instead of comparing the nine meta-learners using a single statistical test, more statistical tests were employed to investigate the classification performances for the same type of classification algorithm with different variants, i.e., intra-variant (subgroup) comparison. Specifically, the subgroups for intra-variant comparison of non-ensemble-based meta-learners are:

- Subgroup 1: Stacked ensembles with decision tree meta-learners (SG-C45, SG-C50, and SG-CART) were compared against each other, and the best performing approach was selected as the representative method for this subgroup;
- Subgroup 2: Stacked ensembles with SVM meta-learners (SG-SVM-Linear, SG-SVM-Radial, SG-SVM-Poly, SG-SVM-Sigmoid) were evaluated, and the top-performing stacked ensemble with SVM meta-learner was chosen;
- Subgroup 3: Stacked ensemble with Naive Bayes meta-learner (SG-NB) was selected without intra-variant comparison;
- Subgroup 4: Stacked ensemble with Feedforward Neural Network meta-learner (SG-NNET) was selected without intra-variant comparison.

After the intra-variant comparisons, four stacked ensembles (each from the four subgroups) were compared, and the best performing stacked ensemble was selected to represent the stacked ensemble with non-ensemble-based meta-learner, i.e., the intra non-ensemble group.

For the intra-variant comparison of stacked ensembles with decision tree meta-learners (Subgroup 1), the Friedman Aligned Ranks test was first conducted. The p-value of the Friedman Aligned Ranks test is 0.6979, which is much greater than 0.05. This suggests that there is no significant difference within the stacked ensembles with decision tree meta-learners, statistically. Since the p-value of the Friedman Aligned Ranks test is much greater than 0.05, it is pointless to conduct the post hoc test to differentiate among the stacked ensembles with decision tree-based meta-learners. For the sake of selecting the best performing stacked ensemble with decision tree meta-learner, the Wilcoxon signed-rank tests were conducted, as shown in Table 8. Based on the tests, SG-CART was compared against SG-C45 and SG-C50, both the null hypotheses are not rejected at $\alpha = 0.05$, which is aligned with the Friedman Aligned Ranks test. Although no significant statistical difference was established, the SG-CART has higher values of ranks (R+) when compared to both SG-C45 and SG-C50. Thus, the SG-CART was

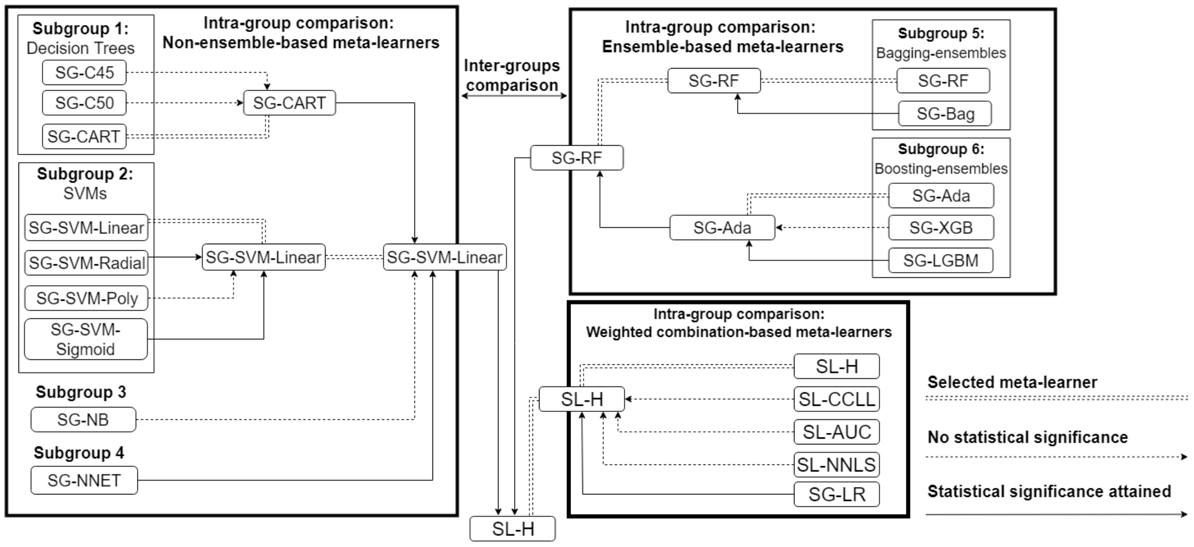

FIGURE 4. Diagram of hierarchical analysis.

TABLE 8. Wilcoxon test - Comparison of decision trees (non-ensemble-based methods).

Comparison	R+	R-	Null Hypothesis ($\alpha = 0.05$)	p-value	Selected
SG-CART vs SG-C45	122.0	87.5	Not Rejected	0.257	SG-CART
SG-CART vs SG-C50	118.0	92.0	Not Rejected	0.314	SG-CART

TABLE 9. Wilcoxon test - Comparison of SVMs (non-ensemble-based methods).

Comparison	R+	R-	Null Hypothesis ($\alpha = 0.05$)	p-value	Selected
SG-SVM-Linear vs SG-SVM-Poly	108	102	Not Rejected	0.463	SG-SVM-Linear
SG-SVM-Linear vs SG-SVM-Sigmoid	140.0	70.5	**Rejected for SG-SVM-Linear**	0.099*	SG-SVM-Linear
SG-SVM-Linear vs SG-SVM-Radial	150.0	60.5	**Rejected for SG-SVM-Linear**	0.048**	SG-SVM-Linear

selected to represent the stacked ensembles with decision tree meta-learners (Subgroup 1).

For the intra-variant comparison of four stacked ensembles with SVM meta-learners (Subgroup 2), the Friedman Aligned Ranks test was conducted. The Friedman Aligned Ranks test has a p-value of 0.0531 that is greater than 0.05, suggesting no significant difference within the stacked ensembles with SVM meta-learners. The Wilcoxon signed-rank tests were employed to select the representative stacked ensemble of Subgroup 2, as shown in Table 9. Based on the pair-wise Wilcoxon signed-rank tests, SG-SVM-Linear has greater R+ values, i.e., higher ranks compared to SG-SVM-Radial, SG-SVM-Sigmoid, and SG-SVM-Poly. Therefore, SG-SVM-Linear was chosen as the candidate from this subgroup. Notice that SG-SVM-Linear has slightly better ranks than SG-SVM-Poly.

Subsequently, the selected representative methods, i.e., SG-CART (Subgroup 1), SG-SVM-Linear (Subgroup 2), SG-NB (Subgroup 3), and SG-NNET (Subgroup 4) competed against each other to represent the stacked ensembles with non-ensemble-based meta-learners (intra non-ensemble group). The Friedman Aligned Ranks test conducted showed a p-value of 0, suggesting a significant statistical difference

TABLE 10. Post hoc test - Comparison of non-ensemble-based methods.

Friedman Aligned Ranks Test = 0 < 0.05			
Comparison	**Null Hypothesis ($\alpha = 0.05$)**	**p-value**	**Selected**
SG-SVM-Linear	Control Method		
SG-NB	Not Rejected	0.989	-
SG-CART	Not Rejected	0.325	-
SG-NNET	**Rejected for SG-SVM-Linear**	0.000***	SG-SVM-Linear

TABLE 11. Wilcoxon test - Comparison of non-ensemble-based methods.

Comparison	R+	R-	Null Hypothesis ($\alpha = 0.05$)	p-value	Selected
SG-SVM-Linear vs SG-NB	120	90	Not Rejected	0.288	SG-SVM-Linear
SG-SVM-Linear vs SG-CART	186	23.5	**Rejected for SG-SVM-Linear**	0.001***	SG-SVM-Linear

within the stacked ensembles with non-ensemble-based meta-learners. A post hoc test with p-value adjusted using Bergman and Hommel's procedure was carried out next. The results are presented in Table 10. As the control method, SG-SVM-Linear performed statistically better than SG-NNET at $\alpha = 0.05$, but when compared to SG-NB and SG-CART, there found no statistical difference. To further analyze, the Wilcoxon signed-rank tests were subsequently conducted [82], [85]. Table 11 presents the Wilcoxon signed-rank tests of SG-SVM-Linear against SG-NB and SG-CART. The results show that the SG-SVM-Linear has higher ranks (R+) than SG-NB and SG-CART. Thus, the SG-SVM-Linear was selected to represent the stacked ensembles with non-ensemble-based meta-learners.

2) INTRA COMPARISON OF STACKED ENSEMBLES WITH WEIGHTED COMBINATION-BASED META-LEARNERS (INTRA WEIGHTED COMBINATION GROUP)

For the intra comparison of stacked ensembles with weighted combination-based meta-learners (intra weighted combination group), the Friedman Aligned Ranks test was first carried out. The p-value of the Friedman Aligned Ranks test

TABLE 12. Post hoc test - Comparison of weighted combination-based methods.

Friedman Aligned Ranks Test = 0.0147 $<$ 0.05

Comparison	Null Hypothesis ($\alpha = 0.05$)	p-value	Selected
SL-H	Control Method		
SL-CCLL	Not Rejected	1.000	-
SL-AUC	Not Rejected	1.000	-
SL-NNLS	Not Rejected	1.000	-
SG-LR	**Rejected for SL-H**	0.016**	SL-H

TABLE 13. Wilcoxon test - Comparison of weighted combination-based methods.

Comparison	R+	R-	Null Hypothesis ($\alpha = 0.05$)	p-value	Selected
SL-H vs SL-CCLL	122.0	88.5	Not Rejected	0.269	SL-H
SL-H vs SL-AUC	128.0	82.5	Not Rejected	0.201	SL-H
SL-H vs SL-NNLS	138.0	71.5	Not Rejected	0.106	SL-H

is 0.0147. This suggests that there is a significant statistical difference at $\alpha = 0.05$ within the stacked ensembles with weighted combination-based meta-learner. A post hoc test with *p*-value adjusted using Bergman and Hommel's procedure was carried out next. The results are presented in Table 12. As the control method, SL-H performed statistically better than SG-LR at $\alpha = 0.05$, and when compared to SL-CCLL, SL-AUC, and SL-NNLS, there found no statistical difference. To further analyze, the Wilcoxon signed-rank test was subsequently conducted [82], [85]. Table 13 presents the Wilcoxon signed-rank test of SL-H against SL-CCLL, SL-AUC, and SL-NNLS. Aligned with the descriptive analysis in Table 7, Table 13 shows that SL-H has a higher ranking (R+) than SL-CCLL, SL-AUC, and SL-NNLS, but despite this, there attained no significant differences. SL-H was selected to represent the stacked ensemble with a weighted combination-based meta-learner.

3) INTRA COMPARISON OF STACKED ENSEMBLES WITH ENSEMBLE-BASED META-LEARNERS (INTRA ENSEMBLE GROUP)

Table 5 shows the ensemble-based meta-learners of two bagging-ensembles (SG-RF and SG-Bag) and three boosting-ensembles (SG-Ada, SG-XGB, and SG-LGBM). The bagging-ensembles and boosting-ensembles are denoted as 'Subgroup 5' and 'Subgroup 6', respectively. The best performing stacked ensemble from each subgroup was selected, i.e., choosing the best performing stacked ensemble with bagging-ensemble meta-learner and the top-performing stacked ensemble with boosting-ensemble meta-learner. Subsequently, the two selected stacked ensembles (each from Subgroup 5 and 6) competed to represent the stacked ensembles with ensemble-based meta-learners.

The Wilcoxon signed-rank test was conducted for pair-wise comparison of stacked ensembles with bagging-ensemble-based meta-learners, i.e., SG-RF and SG-Bag. Based on the result of the Wilcoxon signed-rank test shown

TABLE 14. Wilcoxon test - Comparison of bagging-ensemble-based methods.

Comparison	R+	R-	Null Hypothesis ($\alpha = 0.05$)	p-value	Selected
SG-RF vs SG-Bag	176	34.5	**Rejected for SG-RF**	0.004***	SG-RF

TABLE 15. Post hoc test - Comparison of boosting-ensemble-based methods.

Friedman Aligned Ranks Test = 0 $<$ 0.05

Comparison	Null Hypothesis ($\alpha = 0.05$)	p-value	Selected
SG-Ada	Control Method		
SG-XGB	Not Rejected	0.134	-
SG-LGBM	**Rejected for SG-Ada**	0.000***	SG-Ada

TABLE 16. Wilcoxon test - Comparison of boosting-ensemble-based methods.

Comparison	R+	R-	Null Hypothesis ($\alpha = 0.05$)	p-value	Selected
SG-Ada vs SG-XGB	132	78.5	Not Rejected	0.161	SG-Ada

in Table 14, the null hypothesis is rejected at $\alpha = 0.05$. This shows that SG-RF and SG-Bag are statistically different in terms of classification performance. The R+ and R- represent the rank of SG-RF and SG-Bag, respectively. SG-RF has a higher rank (R+) than SG-Bag. Thus, the SG-RF has a better performance than SG-Bag and was selected as the best performing stacked ensemble with bagging-ensemble meta-learner.

For comparison of stacked ensembles with boosting-based meta-learners (SG-Ada, SG-XGB, and SG-LGBM), the Friedman Aligned Ranks test was first performed. The Friedman Aligned Ranks test has a *p*-value of 0. It means that the classification performances among the stacked ensembles with boosting-based meta-learners are statistically different. To further analyze, a post hoc test with *p*-value adjusted using Bergman and Hommel's procedure was conducted and the results are presented in Table 15. As the control method, SG-Ada performed statistically better than SG-LGBM at $\alpha = 0.05$, and when compared to SG-XGB, there found no statistical difference. For the sake of selecting the best stacked ensemble with boosting-ensemble meta-learner, the Wilcoxon signed-rank test was carried out to compare SG-Ada and SG-XGB, and the results are presented in Table 16. The results show that SG-Ada has a higher ranking (R+) than SG-XGB, but no significant difference was recorded. With a higher R+ value compared to SG-XGB, SG-Ada was selected as the best stacked ensemble with boosting-based meta-learner.

As discussed in the previous paragraphs, SG-RF and SG-Ada were chosen as the best-performing stacked ensemble with bagging-ensemble meta-learner and the top-performing stacked ensemble with boosting-ensemble meta-learner, respectively. To select the better performing stacked ensemble with ensemble-based meta-learner for intra ensemble group, SG-RF and SG-Ada were evaluated using

TABLE 17. Wilcoxon test - Comparison of ensemble-based methods (SG-RF and SG-Ada).

Comparison	R+	R-	Null Hypothesis ($\alpha = 0.05$)	p-value	Selected
SG-RF vs SG-Ada	160	50.5	**Rejected for SG-RF**	0.021**	SG-RF

TABLE 18. Post hoc test - Comparison of weighted combination, ensemble and non-ensemble-based methods.

Friedman Aligned Ranks Test = 0.0627 < 0.10			
Comparison	**Null Hypothesis** ($\alpha = 0.05$)	**p-value**	**Selected**
SL-H	Control Method		
SG-SVM-Linear	Rejected for SL-H	0.035	SL-H
SG-RF	Rejected for SL-H	0.035	SL-H

the Wilcoxon signed-rank test, and the test results are presented in Table 17. Table 17 shows that the null hypothesis is rejected at $\alpha = 0.05$. This indicates that SG-RF and SG-Ada are statistically different in terms of classification performance. Notice that R+ and R- represent the rank of SG-RF and SG-Ada, respectively. SG-RF has a higher rank (R+) than SG-Ada. Thus, SG-RF has a better performance than SG-Ada and was selected to represent the intra ensemble group.

4) INTER-GROUP COMPARISON OF STACKED ENSEMBLES WITH WEIGHTED COMBINATION, ENSEMBLE, AND NON-ENSEMBLE-BASED META-LEARNERS

In the previous intra-group comparisons, the SL-H (from intra weighted combination group), SG-SVM-Linear (from intra non-ensemble group), and SG-RF (from intra ensemble group) were selected as the best stacked ensemble with weighted combination-based meta-learner, the top stacked ensemble with non-ensemble-based meta-learner, and the highest performing stacked ensemble with ensemble-based meta-learner, respectively. In this paragraph, these stacked ensembles with different types of meta-learners are compared against each other.

The Friedman Aligned Ranks test was carried out to compare the multiple stacked ensembles. The Friedman Aligned Ranks test has a p-value of 0.0627, between $\alpha = 0.05$ and $\alpha = 0.10$, indicating that the classification performances are statistically different among the best performing stacked ensembles with different meta-learners at $\alpha = 0.10$. A post hoc test with p-value adjusted using Bergman and Hommel's procedure was conducted. The results are presented in Table 18. As the control method, SL-H performed statistically better than both SG-SVM-Linear and SG-RF at $\alpha = 0.05$.

The Wilcoxon signed-rank tests were carried out to pair-wisely compare SL-H against SG-SVM-Linear and SG-RF, and the results are presented in Table 19. Table 19 shows that the stacked ensemble with weighted combination-based meta-learner represented by SL-H is able to establish statistical differences when compared to the best stacked ensemble with ensemble-based meta-learner (SG-RF) and non-ensemble-based meta-learner (SG-SVM-Linear) at

TABLE 19. Wilcoxon test - Comparison of weighted combination, ensemble and non-ensemble-based methods.

Comparison	R+	R-	Null Hypothesis	p-value	Selected
SL-H vs SG-RF	150	60.5	**Rejected for SL-H** ($\alpha = 0.05$)	0.048**	SL-H
SL-H vs SG-SVM-Linear	140	69.5	**Rejected for SL-H** ($\alpha = 0.10$)	0.093*	SL-H

$\alpha = 0.05$ and $\alpha = 0.10$, respectively. In addition, SL-H recorded better ranks (R+ values) compared to SG-RF and SG-SVM-Linear.

Thus, based on the inter-group comparison of the hierarchical analysis, the stacked ensemble with weighted combination-based meta-learner represented by SL-H has a better performance than stacked ensembles with other types of meta-learners. Notice that the findings are aligned with the reported highest averaged fractional rank and averaged H-measure (Table 7).

C. DISCUSSION

By analyzing the results from the descriptive analysis in Section IV-A and statistical analysis in Section IV-B, a discussion is formulated and presented in this section.

Based on the results as shown in Tables 6 and 7, there is no 'one size fits all' meta-learner, i.e., different meta-learners of stacked ensembles excel in different datasets. In terms of H-measure (Table 7), SL-CCLL has the highest performances in "flare-F", "yeast-1-2-8-9_vs_7", "yeast6", and "abalone19" datasets; SG-SVM-Linear recorded the best results in "ecoli-0_vs_1" and "winequality-white-9_vs_4" datasets; SG-NB has the highest H-measure in "ecoli3" dataset; SG-SVM-Poly has the top performance in "yeast-0-2-5-7-9_vs_3-6-8" dataset; SG-RF excels in "led7digit-0-2-4-5-6-7-8-9_vs_1" and "car-good" datasets. In terms of AUC (Table 6), for example, SL-CCLL has the top performances in "yeast-0-2-5-7-9_vs_3-6-8", "flare-F", and "abalone19" datasets; SG-SVM-Poly has the highest AUC in "led7digit-0-2-4-5-6-7-8-9_vs_1" dataset. Thus, no single meta-learner is better than other meta-learners for every dataset.

As highlighted in Section II-E, the H-measure [61], which overcomes the deficiencies of AUC, is a suitable evaluation metric for imbalanced classification. Thus, based on the results of the average H-measure and average rank computed in Table 7, and the hierarchical analysis conducted based on the H-measure as presented in Section IV-B, the following paragraphs discuss the selection of stacked ensemble's meta-learner for imbalanced classification from two perspectives, i.e., the perspectives of best (single) performing meta-learner and top-performing type of meta-learner.

From the best (single) performing meta-learner perspective, the SL-H recorded the highest average H-measure and average rank values in Table 7. Besides, based on the hierarchical analysis in Section IV-B, the SL-H performed statistically better than other types of meta-learners, i.e., non-ensemble-based and ensemble-based meta-learners of stacked ensembles. The results of descriptive analysis and hierarchical analysis are aligned, i.e., the SL-H has the best overall classification performance on the evaluated imbalanced datasets. This insight is supported by weak evidence as no large margin of superiority is observed. Notice that the SL-CCLL recorded the same highest averaged H-measure as SL-H (Table 7). Besides, as shown in Section IV-B2, the SL-H has higher ranks (R+) than SL-CCLL, SL-AUC, and SL-NNLS. Despite this situation, there attained no significant differences. Nonetheless, the SL-H was selected as the top single meta-learner as it shares the highest averaged H-measure with SL-CCLL, has the highest averaged ranks (Table 7), and is the top-performing method in the hierarchical analysis in Section IV-B.

In terms of the top-performing type of meta-learner in stacked ensemble, the results of the descriptive analysis (Table 7) and statistical analysis evidenced that the weighted combination-based meta-learner is the best performing type of meta-learner when compared to other types such as non-ensemble-based meta-learners (which consists Decision Trees, SVMs, Naive Bayes, and Feedforward Neural Network) and ensemble-based meta-learners (includes boosting and bagging-based ensembles). Specifically, based on the descriptive analysis, the stacked ensembles with weighted combination-based meta-learners (SL-H, SL-CCLL, and SL-AUC) are the top 3 performing stacked ensembles in terms of average H-measure and average rank. Supported by the findings of descriptive analysis and hierarchical analysis in Sections IV-B2 and IV-B4, the stacked ensembles with weighted combination-based meta-learners are statistically better than other stacked ensembles with other types of meta-learners.

In the previous paragraph, we discussed that the weighted combination-based meta-learner is the best performing type of meta-learner. Specifically, the stacked ensembles with weighted combination-based meta-learners are SG-LR, SL-H, SL-AUC, SL-CCLL, and SL-NNLS. Notice that the Logistic Regression meta-learner (in SG-LR) is a very popular meta-learner of a stacked ensemble in imbalanced classification literature. Whereas the SL-H, SL-AUC, SL-CCLL, and SL-NNLS are often coined as Super Learner, i.e., an ensemble consisting of a meta-learner that finds the optimally weighted combination of base learners. For the following discussion, we denote the meta-learner of Super Learner as 'Super Learning-based meta-learner'. Thus, given the popularity of Logistic Regression meta-learner and the distinct naming convention of Super Learner, the following paragraphs discuss the performance improvement if Logistic Regression meta-learner is replaced with Super Learning-based meta-learner, the technical differences between the two, and the possible advantages and shortcomings of Super Learning-based meta-learner.

The percentage of performance improvement (in H-measure) is calculated in Table 20. As an example, with SG-LR as the baseline, the percentage improvement of SL-H is (0.857-0.831)/(0.831) $\times$ 100%, where the H-measure performances of SL-H and SG-LR (acquired from Table 7) are 0.857 and 0.831, respectively. As shown in Table 20, if the Logistic Regression meta-learner is replaced with other Super Learning-based meta-learners, the percentages of improvement (in H-measure) range from 2.5% to 3.1%. Note that the importance of performance improvement varies from case to case. In certain cases, a small improvement of performance is crucial and beneficial. For example, in credit risk prediction, a small improvement of classification performance can result in substantial cost savings for financial institutions [86]; in cancer diagnosis prediction, wrong predictions on actual cancerous patients can cost patients' lives [87] or poor treatment modality.

TABLE 20. H-measure performance improvement with Super Learning-based meta-learner as replacement of Logistic Regression meta-learner in stacked ensemble.

Stacked Ensemble	Percentage Improvement
SL-H	3.1%
SL-AUC	2.8%
SL-CCLL	3.1%
SL-NNLS	2.5%
SG-LR	-

In this paragraph, we discuss the technical differences between Logistic Regression meta-learner and Super Learning-based meta-learner from the perspective of meta-learning function. The function of a meta-learner is important as it returns the final predictions of a stacked ensemble. The functions of Logistic Regression meta-learner and Super Learning-based meta-learner are shown in Eq. (5) and Eq. (11), respectively.

$$h(x) = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_f x_f \tag{11}$$

where:

$\beta_1, \ldots, \beta_f$ = the weights of base learners

f = the number of base learners

Notice that the intercept, β_0 and sigmoid function (6) are required in the function of Logistic Regression meta-learner (5) but are omitted in the function of Super Learning-based meta-learner (11). From the perspective of meta-learning, function (11) finds the best weighted combination of base learners in stacked ensemble if the parameters, $\{\beta_1, \ldots, \beta_f\}$ are optimal, whereas function (5) finds the best sigmoid curve parameterized by z, where z consists of the weighted combination of base learners and intercept, β_0. Given the different components in the meta-learners' functions, intuitively, the finding of an optimally weighted combination of base learners to merge (learn) the base learners' predictions for stacked ensemble (i.e., Super Learning-based meta-learner) is comparatively more sensible.

In this paragraph, we briefly discuss the strengths and shortcomings of the Super Learning-based meta-learner. In general, the strengths of Super Learning-based meta-learner are three-fold. Firstly, Super Learning-based meta-learner has good interpretability as it finds the best weighted combination (linear or non-linear) of base learners as the prediction of stacked ensemble. Secondly, it is easy to use as it requires minimal hyper-parameters tuning. Thirdly, the Super Learning-based meta-learner allows customizable loss function. For instance, the loss function in the H-measure maximizing meta-learner is customized to include the desired H-measure as the metric of maximization. The shortcoming of Super Learning-based meta-learner, in certain configurations, can be originated from its loss function component. As an example, the H-measure maximizing meta-learner in SL-H has a more complex H-measure calculation in the loss function.

V. CONCLUSION

The classification performance of stacked ensemble on imbalanced datasets depends on the choice of stacked ensemble's meta-learner. Based on the literature review, researchers adopted a wide range of meta-learners in stacked ensembles for imbalanced classification. The selection of stacked ensemble's meta-learner from a wide range of choices is not an easy task. In this paper, stacked ensembles with 19 different meta-learners were empirically evaluated on imbalanced datasets. Among the 19 meta-learners of stacked ensembles, a new H-measure maximizing meta-learner of stacked ensemble (SL-H), which has never been explored in the imbalanced classification literature, is first introduced, implemented, and evaluated in this article. The classification performances of stacked ensembles with different meta-learners on imbalanced datasets were evaluated in both the AUC and H-measure metrics. A hierarchical analysis guided by nonparametric statistical tests was conducted to analyze the classification performance of stacked ensemble with different meta-learners from the statistical viewpoint. Based on the descriptive and statistical analyses of imbalanced classification results, the stacked ensembles with weighted combination-based meta-learners, including SL-NNLS, SL-CCLL, SL-AUC, and SL-H, were reported to have better performance than stacked ensembles with ensemble-based meta-learners (SG-RF, SG-Bag, SG-Ada, SG-XGB, and SG-LGBM) and non-ensemble-based meta-learners (SG-C45, SG-C50, SG-CART, SG-NB, SG-NNET, SG-SVM-Linear, SG-SVM-Radial, SG-SVM-Poly, and SG-SVM-Sigmoid). Besides, within the stacked ensembles with weighted combination-based meta-learners, the SL-H, SL-CCLL, SL-AUC, and SL-NNLS were observed to have better performance than the popular stacked ensemble with Logistic Regression as meta-learner (SG-LR). Supported by the empirical imbalanced classification results presented in this paper, some practically useful insights, including the adoption of weighted combination-based meta-learners, were highlighted in selecting stacked ensemble's meta-learner for imbalanced classification.

REFERENCES

[1] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Syst. Appl.*, vol. 73, pp. 220–239, May 2017.

[2] Z. Wang and C. Cao, "Cascade interpolation learning with double subspaces and confidence disturbance for imbalanced problems," *Neural Netw.*, vol. 118, pp. 17–31, Oct. 2019.

[3] S. L. Javan, M. M. Sepehri, M. L. Javan, and T. Khatibi, "An intelligent warning model for early prediction of cardiac arrest in sepsis patients," *Comput. Methods Programs Biomed.*, vol. 178, pp. 47–58, Sep. 2019.

[4] A. Wijaya and R. S. Wahono, "Tackling imbalanced class in software defect prediction using two-step cluster based random undersampling and stacking technique," *Jurnal Teknologi*, vol. 79, nos. 7–2, Nov. 2017.

[5] W. Xing, X. Chen, J. Stein, and M. Marcinkowski, "Temporal predication of dropouts in MOOCs: Reaching the low hanging fruit through stacking generalization," *Comput. Hum. Behav.*, vol. 58, pp. 119–129, May 2016.

[6] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits Syst. Mag.*, vol. 6, no. 3, pp. 21–45, Sep. 2006.

[7] R. T. Clemen, "Combining forecasts: A review and annotated bibliography," *Int. J. Forecasting*, vol. 5, no. 4, pp. 559–583, 1989.

[8] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Inf. Sci.*, vol. 250, pp. 113–141, Nov. 2013.

[9] M. P. Sesmero, A. I. Ledezma, and A. Sanchis, "Generating ensembles of heterogeneous classifiers using stacked generalization," *Wiley Interdiscipl. Rev. Data Mining Knowl. Discovery*, vol. 5, no. 1, pp. 21–34, Jan. 2015.

[10] C.-X. Zhang and R. P. W. Duin, "An experimental study of one- and two-level classifier fusion for different sample sizes," *Pattern Recognit. Lett.*, vol. 32, no. 14, pp. 1756–1767, Oct. 2011.

[11] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, Jan. 1992.

[12] G. Wang, J. Hao, J. Ma, and H. Jiang, "A comparative assessment of ensemble learning for credit scoring," *Expert Syst. Appl.*, vol. 38, no. 1, pp. 223–230, Jan. 2011.

[13] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann, 1993.

[14] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.

[15] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[16] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[17] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, Aug. 2016, pp. 785–794.

[18] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 3146–3154.

[19] G. Batista, D. F. Silva, and R. C. Prati, "An experimental design to evaluate class imbalance treatment methods," in *Proc. 11th Int. Conf. Mach. Learn. Appl.*, Dec. 2012, pp. 95–101.

[20] R. C. Prati, G. E. A. P. A. Batista, and D. F. Silva, "Class imbalance revisited: A new experimental setup to assess the performance of treatment methods," *Knowl. Inf. Syst.*, vol. 45, no. 1, pp. 247–270, Oct. 2015.

[21] P. Branco, L. Torgo, and R. P. Ribeiro, "A survey of predictive modeling on imbalanced domains," *ACM Comput. Surveys*, vol. 49, no. 2, pp. 1–50, Nov. 2016.

[22] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, "Classification with class imbalance problem: A review," *Int. J. Adv. Soft Comput. Appl.*, vol. 7, no. 3, pp. 176–204, 2015.

[23] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 23, no. 4, pp. 687–719, 2009.

[24] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[25] Q. Li and Y. Mao, "A review of boosting methods for imbalanced data classification," *Pattern Anal. Appl.*, vol. 17, no. 4, pp. 679–693, Nov. 2014.

[26] M. J. van der Laan, E. C. Polley, and A. E. Hubbard, "Super learner," *Stat. Appl. Genet. Mol. Biol.*, vol. 6, no. 1, Jan. 2007.

[27] A. I. Naimi and L. B. Balzer, "Stacked generalization: An introduction to super learning," *Eur. J. Epidemiology*, vol. 33, no. 5, pp. 459–464, May 2018.

[28] E. LeDell, M. J. van der Laan, and M. Petersen, "AUC-maximizing ensembles through metalearning," *Int. J. Biostatistics*, vol. 12, no. 1, pp. 203–218, May 2016.

[29] M. F. Kabir and S. A. Ludwig, "Enhancing the performance of classification using super learning," *Data-Enabled Discovery Appl.*, vol. 3, no. 1, p. 5, Jan. 2019.

[30] J. Niestroy, J. Han, J. Luo, R. Zhao, D. E. Lake, and A. Flower, "Prediction of decompensation in patients in the cardiac ward," in *Proc. Syst. Inf. Eng. Des. Symp. (SIEDS)*, 2019, pp. 1–6.

[31] M. L. Williams, W. P. James, and M. T. Rose, "Variable segmentation and ensemble classifiers for predicting dairy cow behaviour," *Biosyst. Eng.*, vol. 178, pp. 156–167, Feb. 2019.

[32] A. Idris and A. Khan, "Churn prediction system for telecom using filter-wrapper and ensemble classification," *Comput. J.*, vol. 60, no. 3, pp. 410–430, Mar. 2017.

[33] M. Ahmed, H. Afzal, I. Siddiqi, M. F. Amjad, and K. Khurshid, "Exploring nested ensemble learners using overproduction and choose approach for churn prediction in telecom industry," *Neural Comput. Appl.*, vol. 32, no. 8, pp. 3237–3251, Apr. 2020.

[34] I. Grenet, K. Merlo, J.-P. Comet, R. Tertiaux, D. Rouquié, and F. Dayan, "Stacked generalization with applicability domain outperforms simple QSAR on *in vitro* toxicological data," *J. Chem. Inf. Model.*, vol. 59, no. 4, pp. 1486–1496, Apr. 2019.

[35] Z. Cao, X. Pan, Y. Yang, Y. Huang, and H.-B. Shen, "The lncLocator: A subcellular localization predictor for long non-coding RNAs based on a stacked ensemble classifier," *Bioinformatics*, vol. 34, no. 13, pp. 2185–2194, Jul. 2018.

[36] F. Akhtar, J. Li, Y. Pei, A. Imran, A. Rajput, M. Azeem, and Q. Wang, "Diagnosis and prediction of large-for-gestational-age fetus using the stacked generalization method," *Appl. Sci.*, vol. 9, no. 20, p. 4317, 2019.

[37] Y. Yuan, K. Ji, R. Sun, K. Ma, Z. Chen, and L. Wang, "An abnormal phone identification model with meta-learning two-layer framework based on PCA dimension reduction," in *Proc. 11th Int. Conf. Mach. Learn. Comput.*, Feb. 2019, pp. 511–515.

[38] I. Tozlu, C. S. G. Öğüdücü, A. Çelebi, S. Kalaycı, and S. Arslan, "Accelerating balance sheet adjustment process in commercial loan applications with machine learning methods," in *Proc. 11th Int. Conf. Electron., Comput. Artif. Intell. (ECAI)*, Jun. 2019, pp. 1–8.

[39] A. Gomaa, S. El-Shorbagy, W. El-Gammal, M. Magdy, and W. Abdelmoez, "Using resampling techniques with heterogeneous stacking ensemble for mobile app stores reviews analytics," in *Proc. Int. Conf. Adv. Intell. Syst. Inf.*, 2020, pp. 831–841.

[40] S. El-Shorbagy, W. El-Gammal, and W. Abdelmoez, "Using smote and heterogeneous stacking in ensemble learning for software defect prediction," in *Proc. 7th Int. Conf. Softw. Inf. Eng.*, 2018, pp. 44–47.

[41] J. Zhou, G. Wang, J. Liu, D. Wu, W. Xu, Z. Wang, J. Ye, M. Xia, Y. Hu, and Y. Tian, "Automatic sleep stage classification with single channel EEG signal based on two-layer stacked ensemble model," *IEEE Access*, vol. 8, pp. 57283–57297, Mar. 2020.

[42] Y. Xia, C. Liu, B. Da, and F. Xie, "A novel heterogeneous ensemble credit scoring model based on bstacking approach," *Expert Syst. Appl.*, vol. 93, pp. 182–199, Mar. 2018.

[43] A. Zhou, K. Ren, X. Li, and W. Zhang, "MMSE: A multi-model stacking ensemble learning algorithm for purchase prediction," in *Proc. IEEE 8th Joint Int. Inf. Technol. Artif. Intell. Conf. (ITAIC)*, May 2019, pp. 96–102.

[44] S.-C. Lin, Y.-C.-I. Chang, and W.-N. Yang, "Meta-learning for imbalanced data and classification ensemble in binary classification," *Neurocomputing*, vol. 73, nos. 1–3, pp. 484–494, Dec. 2009.

[45] H. He, W. Zhang, and S. Zhang, "A novel ensemble method for credit scoring: Adaption of different imbalance ratios," *Expert Syst. Appl.*, vol. 98, pp. 105–117, May 2018.

[46] N. T. Cockroft, X. Cheng, and J. R. Fuchs, "STarFish: A stacked ensemble target fishing approach and its application to natural products," *J. Chem. Inf. Model.*, vol. 59, no. 11, pp. 4906–4920, Nov. 2019.

[47] N. El-Rashidy, S. El-Sappagh, T. Abuhmed, S. Abdelrazek, and H. M. El-Bakry, "Intensive care unit mortality prediction: An improved patient-specific stacking ensemble model," *IEEE Access*, vol. 8, pp. 133541–133564, Jul. 2020.

[48] A. Arora, A. Srivastava, and S. Bansal, "Business competitive analysis using promoted post detection on social media," *J. Retailing Consum. Services*, vol. 54, May 2020, Art. no. 101941.

[49] V. Sobanadevi and G. Ravi, "Handling data imbalance using a heterogeneous bagging-based stacked ensemble (HBSE) for credit card fraud detection," in *Intelligence in Big Data Technologies—Beyond the Hype*. Singapore: Springer, 2021, pp. 517–525.

[50] C. Cao and Z. Wang, "IMCStacking: Cost-sensitive stacking learning with feature inverse mapping for imbalanced problems," *Knowl.-Based Syst.*, vol. 150, pp. 27–37, Jun. 2018.

[51] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, "Machine learning: A review of classification and combining techniques," *Artif. Intell. Rev.*, vol. 26, no. 3, pp. 159–190, Nov. 2006.

[52] W. N. Venables and B. D. Ripley, *Modern Applied Statistics With S*. New York, NY, USA: Springer, 2002.

[53] B. Taylor, *Methodus Incrementorum Directa et Inversa*. London, U.K.: Impensis Gulielmi Innys, 1717.

[54] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. New York, NY, USA: Springer, 2013.

[55] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Boca Raton, FL, USA: Chapman & Hall, 1984.

[56] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Comput. Learn. Theory*, 1992, pp. 144–152.

[57] J.-S. Chou, C.-K. Chiu, M. Farfoura, and I. Al-Taharwa, "Optimizing the prediction accuracy of concrete compressive strength based on a comparison of data-mining techniques," *J. Comput. Civil Eng.*, vol. 25, no. 3, pp. 242–253, May 2011.

[58] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. (2019). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R Package, Version 1.7-3*. [Online]. Available: https://CRAN.R-project.org/package=e1071

[59] R. E. Schapire, "A brief introduction to boosting," in *Proc. 16th Int. Joint Conf. Artif. Intell. (IJCAI)*, vol. 2, 1999, pp. 1401–1406.

[60] N. V. Chawla, "Data mining for imbalanced datasets: An overview," in *Data Mining and Knowledge Discovery Handbook*. Boston, MA, USA: Springer, 2009, pp. 875–886.

[61] D. J. Hand, "Measuring classifier performance: A coherent alternative to the area under the ROC curve," *Mach. Learn.*, vol. 77, no. 1, pp. 103–123, Oct. 2009.

[62] S. Lessmann, B. Baesens, H.-V. Seow, and L. C. Thomas, "Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research," *Eur. J. Oper. Res.*, vol. 247, no. 1, pp. 124–136, Nov. 2015.

[63] D. J. Hand and C. Anagnostopoulos, "A better beta for the H measure of classification performance," *Pattern Recognit. Lett.*, vol. 40, pp. 41–46, Apr. 2014.

[64] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García, "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Multiple-Valued Logic Soft Comput.*, vol. 17, nos. 2–3, pp. 255–287, 2011.

[65] R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, 2020. [Online]. Available: https://www.R-project.org/

[66] J. Abellán and J. G. Castellano, "A comparative study on base classifiers in ensemble methods for credit scoring," *Expert Syst. Appl.*, vol. 73, pp. 1–10, May 2017.

[67] B. Zhu, B. Baesens, and S. K. L. M. vanden Broucke, "An empirical comparison of techniques for the class imbalance problem in churn prediction," *Inf. Sci.*, vol. 408, pp. 84–99, Oct. 2017.

[68] D. Bowes, T. Hall, and J. Petrić, "Software defect prediction: Do different classifiers find the same defects?" *Softw. Qual. J.*, vol. 26, no. 2, pp. 525–552, Jun. 2018.

[69] D. Di Nucci, F. Palomba, R. Oliveto, and A. De Lucia, "Dynamic selection of classifiers in bug prediction: An adaptive method," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 1, no. 3, pp. 202–212, Jun. 2017.

[70] K. Hornik, C. Buchta, and A. Zeileis, "Open-source machine learning: R meets weka," *Comput. Statist.*, vol. 24, no. 2, pp. 225–232, May 2009.

[71] L. Mouselimis. (2019). *Kernelknn: Kernel K Nearest Neighbors, R Package Version 1.1.0*. [Online]. Available: https://CRAN.R-project.org/package=KernelKnn

[72] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[73] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou, M. Li, J. Xie, M. Lin, Y. Geng, and Y. Li. (2019). *XGBoost: Extreme Gradient Boosting. R Package Version 0.90.0.2*. [Online]. Available: https://CRAN.R-project.org/package=xgboost

[74] S. G. Johnson. (2014). *The NLopt Nonlinear-Optimization Package*. [Online]. Available: http://ab-initio.mit.edu/nlopt
[75] C. Anagnostopoulos and D. J. Hand. (2019). *Hmeasure: The H-Measure and Other Scalar Classification Performance Metrics. R Package Version 1.0-2*. [Online]. Available: https://CRAN.R-project.org/package=hmeasure
[76] E. Polley, E. LeDell, C. Kennedy, and M. van der Laan. (2019). *Super-learner: Super Learner Prediction. R Package Version 2.0-27-9000*. [Online]. Available: https://github.com/ecpolley/SuperLearner
[77] M. Kuhn and R. Quinlan. (2020). *C50: C5.0 Decision Trees and Rule-Based Models, R Package Version 0.1.3.1*. [Online]. Available: https://CRAN.R-project.org/package=C50
[78] T. Therneau and B. Atkinson. (2019). *rpart: Recursive Partitioning and Regression Trees, R Package Version 4.1-15*. [Online]. Available: https://CRAN.R-project.org/package=rpart
[79] A. Peters and T. Hothorn. (2019). *ipred: Improved Predictors, R Package Version 0.9-9*. [Online]. Available: https://CRAN.R-project.org/package=ipred
[80] E. Alfaro, M. Gámez, and N. García, "adabag: An R package for classification with boosting and bagging," *J. Stat. Softw.*, vol. 54, no. 2, pp. 1–35, 2013.
[81] G. Ke, D. Soukhavong, J. Lamb, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Y. Yan, M. Corporation, Dropbox, Inc., J. Loden, D. Daeschler, G. Rodola, A. Ferreira, D. Lemire, V. Zverovich, and IBM Corporation. (2020). *Lightgbm: Light Gradient Boosting Machine, R Package Version 3.1.0*. [Online]. Available: https://CRAN.R-project.org/package=lightgbm
[82] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 4, pp. 463–484, Jul. 2012.
[83] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.
[84] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci.*, vol. 180, no. 10, pp. 2044–2064, May 2010.
[85] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera, "EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling," *Pattern Recognit.*, vol. 46, no. 12, pp. 3460–3471, Dec. 2013.
[86] A. I. Marqués, V. García, and J. S. Sánchez, "Two-level classifier ensembles for credit risk assessment," *Expert Syst. Appl.*, vol. 39, no. 12, pp. 10916–10922, Sep. 2012.
[87] S. Fotouhi, S. Asadi, and M. W. Kattan, "A comprehensive data level analysis for cancer diagnosis on imbalanced data," *J. Biomed. Informat.*, vol. 90, Feb. 2019, Art. no. 103089.

SENG ZIAN received the B.Sc. degree (Hons.) in statistics from the National University of Malaysia, in 2014. He is currently pursuing the Ph.D. degree with the Faculty of Computer Science and Information Technology, University of Malaya, Malaysia. His research interests include data science, business analytics, information retrieval, and machine learning.

SAMEEM ABDUL KAREEM received the B.Sc. degree (Hons.) in mathematics from the University of Malaya, Malaysia, in 1986, the M.Sc. degree in computing from the University of Wales, Cardiff, U.K., in 1992, and the Ph.D. degree in computer science from the University of Malaya, in 2002. She is currently a Professor with the Department of Artificial Intelligence, Faculty of Computer Science and Information Technology, University of Malaya. She has published more than 100 peer-reviewed articles in journals and conference proceedings. Her research interests include biomedical informatics, machine learning, data mining and intelligent techniques, ontologies, and image processing.

KASTURI DEWI VARATHAN received the Ph.D. degree in computer science from the National University of Malaysia. She has served as a Visiting Scientist with the Information Systems Research Group, Faculty of Informatics, University of Lugano, Switzerland, and a Research Fellow with the Institute of Visual Informatics, National University of Malaysia. She is currently a Senior Lecturer with the Faculty of Computer Science and Information Technology, University of Malaya, Malaysia. She has published in several high-rank journals and conferences. Her main research interests include data analytics and information retrieval. She was a recipient of the Prestigious Leadership in Innovation Fellowship Award by the U.K. and Malaysian Government.